2015

Language Processing and the Artificial Mind: Teaching Code Literacy in the Humanities

Jerry Scott Weltman
Louisiana State University and Agricultural and Mechanical College

LANGUAGE PROCESSING AND THE ARTIFICIAL MIND: TEACHING CODE LITERACY IN THE HUMANITIES

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Arts

in

The Interdepartmental Program of Linguistics

by
Jerry Scott Weltman
B.A., University of Texas, 1984
M.S., Stanford University, 1985
Ph.D., Louisiana State University, 2013
August 2015

ACKNOWLEDGEMENTS

My idea for this case study began when Lisi Oliver (Houston Area Alumni Professor of Linguistics and English at LSU) suggested I teach a senior-level special topics course on language processing and the development of artificial languages, areas related to previous work on my computer science PhD (LSU, 2013). It is bittersweet to have developed the course and completed this Linguistics MA thesis. Just nine days before my MA defense, Lisi tragically died in a road accident. We miss her terribly.

As with my recent dissertation, I thank the faculty and staff at Louisiana State University for their enthusiastic support of my interdisciplinary studies. I owe a particular debt to Rafael Orozco, chair of the Linguistics program, who graciously stepped in last minute as chair of my committee after Lisi's passing. I am especially grateful to committee member David Kirshner from Math Education, who spent extra time with me to help me solidify some key ideas in this thesis. I also very much appreciate the help and support of my other committee members, Yejun Wu from Library and Information Science, and Jon Cogburn from Philosophy. Thanks go, as always, to Michael Hegarty for his generous mentorship.

In the often impersonal environment of a large university, I received warmth and kindness from Ashley Thibodeaux and Margaret Jo Borland, both of whom went way beyond their respective duties to help me over the various bureaucratic hurdles needed to do this work.

I gratefully acknowledge my good friend Mike Boechler for having an interest in my work and helping me brainstorm ideas for this project. And of course, I must express my heartfelt appreciation to the LSU students of the fall class of Ling/English 4310 for their enthusiastic participation in an experimental new humanities course. The class and I were very fortunate to

have a special guest-poet, LSU's Laura Theobald, to enrich the material on conversational agents.

Finally, I lovingly thank my wife Sharon and children Alex and Elizabeth. All three helped me enormously throughout this project with ideas for course material and classroom activities. I could not do without their constant love and support.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii

TABLE OF CONTENTS v

ABSTRACT viii

CHAPTER 1: INTRODUCTION 1
Statement of the Problem 1
Research Question 4
Theoretical Framework 6
Significance of the Project 8
What Follows 10

CHAPTER 2: RELATED WORK 13
Introduction 13
Computational Thinking and Code Literacy 14
University Programs that Combine Humanities and Computer Science 16
Computational Linguistics and NLP 18
Outreach for Education in Computer Science 21
Invented Languages in the Classroom 23

CHAPTER 3: COURSE CONTENT 25
Introduction 25
Unit 1: The Turing Test: Language as a measure of intelligence 27
Unit 2: Computational Thinking: How computer programs work 30
Unit 3: Computational Semantics: Why language is difficult for computers to process 38
Introduction 39
Eight Challenges for NLP 42
Analysis of a Children's Story 50
Unit 4: Statistical Natural Language Processing: How Siri and Google work 50
Unit 5: Computers and Common Sense: Teaching computers about human experience 54
Introduction 54
Scripts (1977) 56
Cyc (1984) 57
Open Mind Common Sense (2002) 59
The Human Experience Project (2013) 61

Unit 6: AI in Popular Culture 65
Introduction 65
HAL from 2001, A Space Odyssey 66
Lt. Commander Data, from Star Trek: The Next Generation 72
Unit 7: Group presentations on invented languages or AI-related literature 76
Conclusion 77

CHAPTER 4: COURSE EVALUATION AND RECOMMENDATIONS 79
Introduction 79
Description of the Surveys 80
Survey Results 80
Unit 1: The Turing Test 82
Unit 2: Computational Thinking 82
Unit 3: Computational Semantics 83
Unit 4: Statistical Natural Language Processing 83
Unit 5: Collecting Common Sense 84
Unit 6: AI in popular culture 84
Unit 7: Presentations on Invented Languages 85
Effectiveness of Class 85
Conclusion 88

CONCLUSION 89

REFERENCES 92

APPENDIX 1: COURSE SYLLABUS 98

APPENDIX 2: COURSE SCHEDULE 99

APPENDIX 3: UNIT 1: THE TURING TEST 103
Unit Objectives 103
Results from Playing the Imitation Game 103
Unit 1 Exercises 106

APPENDIX 4: UNIT 2: COMPUTATIONAL THINKING 109
Unit Objectives 109
Python Programs 109
Unit 2 Exercise 1 118
Unit 2 Exercise 2 120

APPENDIX 5: UNIT 3: STORY UNDERSTANDING 122
Unit Objectives 122
Introduction 122
Grice's Cooperative Principles 122
Example of implicatures 123
Analyses of Sentences 124
Sentence 1: 124
Sentence 2: 126
Sentence 3, Clause 1: 129
Sentence 3, Clause 2: 131
Sentence 4: 135
Sentence 5: 138
Unit 3 Computational Semantics: Exercises 141
Unit 3 Computational Semantics: Group Project 142

APPENDIX 6: UNIT 4: STATISTICAL NLP 144
Unit Objectives 144
Unit 4 Exercises 144

APPENDIX 7: UNIT 5: COLLECTING COMMON SENSE 146
Unit Objectives 146
Unit 5 Exercise 1 on Scripts, Cyc, and OMCS 146
Unit 5 Exercise 2 on HXP 147

APPENDIX 8: UNIT 6: AI IN POP CULTURE 148
Unit Objectives 148
Unit 6 Exercise 148

APPENDIX 9: UNIT 7: INSTRUCTION FOR ORAL PRESENTATIONS 150

APPENDIX 10: SURVEYS 152
Survey 1: 152
Survey 2 155
Survey 3 156

APPENDIX 11: INSTITUTIONAL REVIEW BOARD APPROVAL 158

VITA 159

ABSTRACT

Humanities majors often find themselves in jobs where they either manage programmers or work with them in close collaboration. These interactions often pose difficulties because specialists in literature, history, philosophy, and so on are not usually "code literate." They do not understand what tasks computers are best suited to, or how programmers solve problems. Learning code literacy would be a great benefit to humanities majors, but the traditional computer science curriculum is heavily math oriented, and students outside of science and technology majors are often math averse. Yet they are often interested in language, linguistics, and science fiction. This thesis is a case study to explore whether computational linguistics and artificial intelligence provide a suitable setting for teaching basic code literacy. I researched, designed, and taught a course called “Language Processing and the Artificial Mind.” Instead of math, it focuses on language processing, artificial intelligence, and the formidable challenges that programmers face when trying to create machines that understand natural language. This thesis is a detailed description of the material, how the material was chosen, and the outcome for student learning. Student performance on exams indicates that students learned code literacy basics and important linguistics issues in natural language processing. An exit survey indicates that students found the course to be valuable, though a minority reacted negatively to the material on programming. Future studies should explore teaching code literacy with less programming and new ways to make coding more interesting to the target audience.

CHAPTER 1: INTRODUCTION

Statement of the Problem

Humanities majors typically avoid science, technology, engineering, and math (STEM), and they often graduate with little or no knowledge of computer programming. While in most non-STEM careers an ignorance of calculus or physics is no great handicap, a lack of knowledge about computer programming is. Programming is ubiquitous. Humanities professionals with degrees often find themselves close colleagues with a programmer when they need to analyze data or manage a web site. Many a business owner or employee, once a humanities major, finds him or herself the reluctant manager of a few programmers. The lawyer, banker, advertising executive, hotelier, and sales manager – all typical landing spots for humanities graduates – have frequent need to collaborate with programmers to create or build specialized business applications. And of course the technical writer, almost certainly an English major, deals with programmers every day converting tech-speak into understandable technical design documents and metaphor-rich user manuals.

It's not just the usual separation of concerns between colleagues. Neither is it just the usual lack of understanding among different professions. All businesses have people pigeon-holed into professional areas. The difference is that there is a greater presence of programmers working with and for people who don't understand the world of programming. They don't know what sorts of tasks computers are good at, how programmers solve problems, and what sorts of results are reasonable or unreasonable. Even leaving out the quite common situation of non-programmers managing programmers, we still have the even more common situations of close collaborations between techies and non-techies. Software professionals refer to non-techies as "end users" or "subject matter experts (SMEs)." A huge part of the software development

process involves programmers working closely with end users and SMEs to define the requirements and needs of a particular business area. Communication problems between the disparate disciplines are so common and costly that many books and articles have been written from the computer programmer point of view to help the programmer extract adequate information from the SME (for example, Courage and Baxter, 2005; Sharon, 2012). Websites for programmers post absurdly obvious advice such as "make a check list" and "learn how to really, really listen!"[1]

Instead of teaching computer science majors, typically not great communicators, how to talk to SMEs, I suggest that it makes more sense to teach humanities majors, *communicators par excellence*, about algorithms and programs so that when they are managers, SMEs, or tech writers, they will more effectively *interface* with programmers. Supporting this goal, there are several web sites with names like "Talk like a Programmer"[2] and "20 Hilarious Programming Jargon Phrases You Should Use When Talking to An Engineer."[3] These are clearly aimed at making non-technical hiring managers or business owners more comfortable with the jargon of computer programming so that they appear less ignorant when they have to interact with a cocky programmer. While learning the ever-changing list of hot technical buzz words is probably a good idea, it doesn't get at the more fundamental problem of knowing something about programming. To get this knowledge, one simply needs to do some programming.

But teaching humanities majors how to program faces a significant hurdle. Many humanities majors do not like math,[4] and from the start, computer science education assumes an underlying

[1] http://blog.smartbear.com/requirements/how-to-interview-users-to-find-out-what-they-really-want/
[2] https://generalassemb.ly/education/talk-like-a-programmer
[3] http://www.businessinsider.com/20-hilarious-programming-jargon-phrases-you-should-know-when-talking-to-engineers-2012-7
[4] Anecdotal evidence given in Tobias (1995), and formal studies by Preston (1986) and LeFevre (1992) indicate that students choose non-STEM majors to avoid math not necessarily through lack of ability but rather through lack of confidence.

mathematical foundation. Early programming lessons often begin with detailed definitions of integers and floating point numbers. First programing exercises are often math-oriented problems, such as finding the average of a sequence of numbers or printing the circumference of a circle.

The field of computer science itself is rooted deeply in mathematical computations. The history of computers begins with Charles Babbage, who in 1822 drew plans for a steam-driven Difference Engine to compute tables of numerical calculations. He later designed his more general purpose Analytical Engine. While neither machine was ever built, Babbage's long time correspondent and friend Ada Lovelace, daughter of the poet Lord Byron, not only wrote instructions for computing a complex sequence of numbers on Babbage's planned Analytic Engine, but she also recognized the potential to program Babbage's machine for non-mathematical applications such as composing music. Thus, the first computer programmer was arguably a woman (Toole, 1996). The word "computer" itself initially referred to a person, often female, who did computations in factory-like drudgery to aid in scientific research. When the field of computer science was born in the 1950s, it was mostly considered a tool for mathematics. (Denning, 2000). Although it is now its own discipline, computer science remains highly math-oriented, and computer science majors are normally required to take at least three semesters of advanced math as well as other math-oriented theory.

Being comfortable with math is only one of the many stereotypes of a programmer. More insidious ones conjure up the image of male-dominated "unwashed nerds" not interested in the arts or literature, and these foster an antagonistic divide between computer science and humanities (Moore-Coyler, 2015). Indeed, while participation among girls is gaining in many of the STEM fields, girls continue to lose ground in computer science (Bidwell, 2015). Although

there are no conclusive reasons for the antipathy of girls to computer science, the scholarship cites many possibilities: girls don't have role models in the field; they lack confidence in their technical skills; they feel derided by their male classmates. In addition, teaching techniques are impersonal and do not foster an inclusive environment (Barker, et al., 2005; Moore-Coyler, 2015). Given these problems, aversion to math would be one more reason to drive away the more than two-thirds of humanities students who are female[5] from any interest in learning computer programming.

Research Question

> What is an effective one-semester approach for teaching basic code literacy to humanities majors with no background in programming and who may even have a negative attitude towards computer programming?

As demonstrated in the next chapter, my research question has not been addressed before. The academic goal of teaching computer science in the humanities has only been recently taken up, and the few current curricular approaches vary widely in terms of the amount of computer science expertise taught to non-majors. Given the newness of the question and the lack of comparable studies, my project is an exploratory case study in which I develop the curriculum and teach a one-semester course.

Exploratory case study methods used in this study are appropriate when the existing knowledge base is poor and available literature provides no current framework or working hypotheses. The researcher's goal in such studies is to explore hypotheses of potential relevance to the domain of interest with a view to establishing more firm foundations for future inquiry (Yin, 2014). What is to be explored here is twofold: the types of computational linguistics and AI subject material that may be of interest and comprehensible to humanities students, and

[5] http://www.npr.org/blogs/money/2014/10/28/359419934/who-studies-what-men-women-and-college-majors

whether this material provides a suitable setting for teaching basic code literacy. This case study will present the teaching material, explain why this material was developed, and describe qualitative results from teaching it. At this stage of inquiry, the questions are broad. Will humanities students even be interested enough to register for a course like this? Will they stay in it for the duration? What aspects of the course material could be expanded, reduced, or replaced?

In my experience, although some humanities majors hate math, they almost certainly love language, literature, history, and/or cultural studies. They may not be interested in science, but they often love science fiction, which frequently includes an exploration of artificial intelligence (AI). Thus, the proposition for my case study is that students would be interested in learning code literacy if it applies to natural language processing and AI. My course teaches simple algorithms, program control structures, pattern matching, and the formidable challenges that AI programmers face when trying to create machines that understand natural language. An impediment for this type of course is that much of existing AI educational material assumes a background in formal logic and contains intimidating mathematical notation. In order to attract humanities majors, my course uses no math, and I develop much of the material specifically for the target students. The course starts with a gentle introduction to algorithms and programming because basic code literacy is necessary for students to grasp the difficulties involved in computer processing of natural language. Also important to understanding these difficulties is a familiarity with semantics and pragmatics, and so the course considers how a computer program might treat these important linguistics subfields. Finally, since humanities majors are often fascinated with invented languages from literature and movies, the course gives students a taste of phonology and syntax by surveying the structure of invented languages such as Esperanto, Klingon, and Elvish. This study of invented languages rounds out this course on code literacy

because students learn the difference between computer code (an unambiguous construct similar to a math formula) and constructed languages for communication (which while artificial, share almost all of the linguistic properties of natural languages).

After teaching the course once, I believe that it largely achieved my stated goal; student performance on exams indicates that they understood the material. But there is room for improvement. While a class survey indicates that most of the students had a favorable or neutral opinion of the programming lessons, negative comments indicate that a large minority of students felt the programming material was too complex and boring. As might be expected of humanities majors, they much preferred the material on general linguistics issues, and they really liked the unit on invented languages. In future versions of this class, I would tap more into students' interest in literature by doing small programming exercises to analyze the text of novels, speeches, and social media.

Theoretical Framework

The computer science pioneer Alan Turing (1912-1954) recognized almost as soon as general purpose computers became available in the 1940s that computer programs could extend far beyond number crunching. In his groundbreaking paper, "Computing Machinery and Intelligence" (1950), Turing predicted that machines would one day be considered "intelligent." Interestingly, his yardstick to measure intelligence was the ability to have a natural conversation. That is, for this extraordinary mathematician and computation theorist, language ability – not math – was the measure of intelligence.

Given the importance of language in Turing's thesis, I reasoned that working with issues of AI and natural language processing (NLP) would be an excellent way to motivate humanities students to learn about computation and programming. Modern programming languages like

Python hide many of the tedious details and math underpinnings of older programming languages. Nevertheless, the vast majority of computer science pedagogy starts with math operations. However, with Python the student can immediately work on non-math applications including programs that play around with language and text analysis. So in a sense, teaching programming via NLP is returning to what Turing saw as one of the most exciting capabilities of a computer.

The pedagogical method for this course is related to content-based instruction (CBI), a teaching approach widely used in learning a foreign language. In the CBI approach, the student is motivated to learn the vocabulary and structure of a second language by stimulating content (Grabe & Stoller, 1997, Brinton, Snow, & Wesche, 1989). To some extent, learning the structures and syntax of programming is similar to learning new constructs in a foreign language. By providing a clear motivation and context for how a computer program processes language, humanities majors learn about algorithms, key Python vocabulary words, and general concepts such as "function," "variable," and "parameter" – mathematical terms that can be presented within the context of language understanding rather than math. After students learn how a program works and what sort of information is needed to simulate language understanding, they have an appreciation for the problems of NLP, and issues in semantics and pragmatics become even more relevant. At the same time, students can look at AI in science fiction with a more critical eye. The CBI approach also motivates students to learn about syntax, phonetics, phonology, and socio-linguistics as they try to formulate sentences in Klingon or Elvish, or to describe how the language is relevant to the invented culture.

Significance of the Project

Students of the humanities graduate with excellent analytical skills. They are well rounded in arts and literature, and they are ready to pursue a variety of careers. Knowledge of computer programming will make them able to deal with a computational dimension that so many careers now demand. This project offers a new way of teaching this critical computing knowledge, tailored specifically for the literature-loving person who has an aversion to traditional STEM material. Students are taught enough programming to understand what types of problems are suitable for computation, what programs look like, and how computational processes use data to accomplish the programmers' goals.

Humanities programs, including digital humanities and the linguistics subfields of corpus linguistics, semantics, and pragmatics, would benefit by having a new way to motivate their students to learn about computer programming and NLP. Let's start with digital humanities (DH), a field which integrates humanities and computers with the goal of expanding world-wide interest and access to literature, history, languages, religion, and art. DH is a burgeoning field of study, and many universities are establishing new programs, defining curricula, and setting requirements for certificates of study. Currently, programming courses are often optional for DH students, and the only courses available are related to computer science and engineering. A programming course tailored to the interests of DH students, rather than the typical math-oriented instruction, would help establish code literacy as a core component of a DH program. The result will be a stronger, more cohesive DH discipline.

With regard to linguistics, corpus linguistics is most in need of a course in code literacy. In this subfield of linguistics, students analyze sentence patterns and vocabulary of large bodies of text. For example, they might count the number of occurrences of vocabulary words or active

sentences versus passive sentences. Those who do not automate this analysis through computer programming are basically stuck in the 19th century, making index cards and manually counting words and phrases. Even if students succeed in identifying trends, they are not taking full advantage of the data. With a computer program, they can effortlessly look for many types of patterns, examine far more text, and graphically display results in various charts and graphs that might reveal surprising new connections.

Besides corpus linguistics, students of the core linguistics fields of semantics and pragmatics will develop a fuller appreciation of these theories when they try to imagine using them for AI. Semantics is the study of the meaning of words and phrases. Part of this field includes analysis of lexical relationships such as *synonymy* (e.g., car and auto are synonyms) and *hypernymy* (e.g., bird is a hypernym of sparrow). An NLP computer program needs access to these meanings and lexical relationships in order to look for general patterns in a text. For example, with hypernym data about sparrows, a program could conclude that a sentence with "sparrow" as a subject is also a sentence about birds, which would then allow the program to make connections with other sentences about birds. Being able to make this type of general connection is critical to simulating intelligence in a computer. Now let's look at pragmatics, the study of how sentences are interpreted in context. Pragmatics analyzes conversational conventions, pre-existing discourse information, intentions, background knowledge, and other relevant information that helps us interpret a sentence. As an example of how pragmatic theories can help AI, the theory of *presupposition* identifies information that is taken for granted when certain linguistic triggers occur. A trigger, such as the verb *stop* in "Max has stopped going to the gym," usually allows us to infer that Max used to go the gym. That is, X stopped Y presupposes that X used to do Y. Or X found out about Y presupposes that Y must be true, even if the text does not specifically assert

Y. Thus, given precise data about various presupposition triggers, a computer program could make the same types of presuppositional inferences that humans make. Another pragmatic theory, *implicature*, involves how we infer what others are saying based on what they say indirectly. For example, if John asks Mary on a date, and Mary replies, "I'm busy," John understands that Mary has rejected him. How is a computer to understand that "I'm busy" or "I have to wash my hair" means "no" in some situations? The NLP program would need to have a model of human interactions, intentions, expected responses, and world knowledge to make the same types of implicature-driven inferences that humans seems to effortlessly make. These types of inferences seem to be critical if we are to have computers that can understand stories, summarize text, and answer questions about a text.

A wider impact of this work has to do with the general goal of having more code-literate citizens. Learning about programming removes some of the mystery about what happens behind the scenes when people work with a computer. Just as learning the secret to a magician's trick takes away the magician's power to amaze, understanding programmers' tricks to implement NLP makes us more sophisticated consumers of technology and more knowledgeable in the world of technology. It changes our role from passive receiver of technology to active participant, knowledgeable trouble-shooter, and creative collaborator.

What Follows

Chapter 2 describes related work on integrating humanities and computer science. The initiatives are just beginning in this area, and I examine various programs, from complete integration in a joint computer science/humanities major to a gentle exercise in interdisciplinary collaboration. My project falls in the middle, an individual class devoted to teaching basic code literacy.

Chapter 3 describes in detail each of the seven units of the course. The first unit is on Alan Turing and artificial thinking. The class plays Turing's "Imitation Game" as a fun introduction to Turing's provocative ideas on machine intelligence, followed by an exposition of Turing's life and a reading and discussion of Turing's seminal paper on language as a measure of machine intelligence (Turing, 1952). The next unit is a basic course in programming in Python where students learn about variables, loops, and list processing, culminating in a project to create a "chatbot" program that simulates intelligent conversation. With this basic understanding of programming, students are able to appreciate the next three units on the challenges of programming computers with the ability to processing language at a human level of intelligence. These three units analyze what makes language so difficult for a computer to understand, and they are composed of lectures on statistical NLP, semantics, pragmatics, and the intractable problem of programming computers with common sense. To reinforce their understanding of these problems, students are asked to analyze sentences of a simple children's story in terms of what sorts of linguistic theories and commonsense knowledge would have to be programmed into the computer to simulate understanding the story. The sixth unit applies the material from the previous units to analyze the language capabilities exhibited by AI machines in popular culture. The final unit shifts gears to discuss various linguistic topics involved with invented languages.

Chapter 4 presents some results of teaching this course. It examines student surveys, summarizes the results, and then discusses my own evaluation of the effectiveness of each unit in accomplishing the stated goals. The class was registered to the maximum students—twenty, and all the stayed for the duration and 85% made an A or B. While the results of the exams and the class surveys indicate that students reached some level of code literacy, some of the students did

not feel connected to the material, particularly some of the details of computer programming. The next time the course is taught, I would recommend more programming exercises that tap into the students own interests in analyzing and interpreting text. For example, students could write programs to identify the most frequent words used in one of their previous papers, looking for overused vocabulary. Alternatively, I would recommend exploring whether some of the details on variables and functions could be deleted and while still allowing enough code literacy to understand the issues of natural language processing.

CHAPTER 2: RELATED WORK

Introduction

The main goal of this project is to teach humanities students about computer programming and the difficulties involved in programming computers to understand language. This section reviews several initiatives that combine computer programming with humanities. I begin with the theme most central to my project: teaching computational thinking and code literacy – not so that students become great programmers, but so that they understand programming and are more active participants in technological solutions. Next, I look at university programs that combine humanities and computer science. I first discuss Stanford University's new joint CS/Humanities major as well as another recently introduced elective course at Stanford that pairs computer science students with humanities. Then I discuss a new class at Columbia University that teaches programming for the liberal arts. This section also considers how computing is taught within digital humanities programs. I show that my class covers a curriculum space not addressed by any of these programs. So new is this approach that there are no existing explanatory frameworks as yet to theorize it.

I then make a slight turn away from the humanities to computational linguistics. This increasingly popular sub-field of computer science and linguistics is highly technical and based more on math than traditional linguistic analysis. Nevertheless, it is at the heart of virtually all computer applications that deal with language, such as Google searches, machine translation, and voice recognition. As such, it is important for anyone studying NLP to understand the mathematical techniques and pattern-matching tricks that make these applications appear so intelligent. Unlike virtually all computational linguistics material, which delves deeply into

probability theory, my material presents just enough of the concepts for students to understand the basic statistical approach.

Next, we look briefly at recent outreach programs to broaden computer science education towards girls, significant to my project since most humanities majors are female. There are many well-known barriers to attracting girls to programming, and the typical dislike of math among humanities majors exacerbates the problem. I argue that the language-centric approach to programming may be a way to bring more girls into the field.

Finally, the last section focuses on my secondary goal, that of attracting more students to study linguistics. I cite other schools that leverage the popularity of invented languages to teach linguistics topics, mostly phonology and linguistic typology. These are full semester courses and cover the material in much more detail than the broad survey of my class; nevertheless, the popularity of these courses shows that constructed languages are fruitful approaches to draw in linguistics students.

Computational Thinking and Code Literacy

Jeannette M. Wing popularized the term computational thinking to refer to the way computer scientists solve problems by breaking larger problems into smaller ones, defining objects at different levels of abstraction, and specifying formal rules that govern the behavior of objects at each level. She asserts that "computational thinking is a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (Wing, 2006). Unfortunately for humanities students, many of the techniques described by Wing involve math. For example, Wing's tutorial on

computational thinking[6] focuses on large data analysis, mathematical modeling, and efficiency of algorithms using mathematically sophisticated notation that are likely off-putting to many humanities students.

Tasneem Raja takes a more accessible approach to computational thinking (Raja, 2014). She describes it as the ability to imagine how any process – creating recipes, planning a party, helping fire fighters locate fire hydrants – can be improved by decomposing big steps into smaller ones. It includes deciding which steps must occur before others as well as procedures for testing each step. More importantly, it includes an understanding of which steps are suitable for automation, and this involves an understanding of how a computer program works, what Raja calls "code literacy." One key point for Raja is that learning to write professional-quality code requires a lot of education and experience, and many people find coding to be tedious. She stresses that the important thing is not to be able to write code but rather to understand what code can do. Code literacy contrasts with the older term of "computer literacy," which means the ability to interact with a computer and use standard programs such as a word processor, a spreadsheet, or a search engine. A code-literate person is more of an equal partner with the programmer, is able to guess how the program is working, can ask knowledgeable questions to a programmer, and can make practical suggestions for improvement. Code literacy also means being able to imagine how problems might be solved through computational processes and recognizing what types of obstacles must be overcome in order to make a solution amenable to computation.

My course follows directly from Raja's definition of code literacy. It's not about students becoming programmers but rather about getting enough coding experience so that students can

[6] https://undergrad.stanford.edu/academic-planning/majors-minors/joint-majors-csx

become more effective understanding computer-based technology, and consequently become more effective at whatever career they choose.

University Programs that Combine Humanities and Computer Science

Recognizing that the world needs a combination of skills in both computer science and humanities, Stanford University started the CS+X[7] initiative in 2014. It is a joint degree integrating computer science and humanities to allow students to acquire skills in "separate but mutually galvanizing fields of study."[8] While not quite a double major, the joint degree requires 80% of the core classes of both disciplines. Thus, rather than finding common ground between two fields, it basically requires students to become experts in both, and students expect to spend five or six years to complete the course work. In contrast to this intense program of study where students become equally skilled in two disciplines, my project is a single course that teaches the fundamentals of programming in order to make humanities majors more effective in communicating with programmers.

In another initiative to bridge the divide between CS and humanities, Stanford introduced a single elective course, "Literature and Social Online Learning" that puts students from both disciplines together into one classroom to collaborate on small projects (Beacock, 2014). For their projects, the students create websites to let online users explore and appreciate literature. Thus, the humanities student has the role of subject matter expert who has to work closely with a programmer. The focus of this elective course is on communication across disciplines, not cross-training. While this Stanford elective course fosters admirable interdisciplinary communication,

[7] "CS" of course stands for computer science. "X" is a variable that can stand for any one of a number of humanities majors such as English, History, Classics, and Philosophy.
[8] https://undergrad.stanford.edu/academic-planning/majors-minors/joint-majors-csx

my project requires that students actually learn about programming so that they become more knowledgeable collaborators.

Columbia University introduced an experimental interdisciplinary course in the spring semester of 2015 called "Computing in Context." This course has a very similar goal to my project. One of its course developers, Adam Cannon describes it as "an introduction to computing for the liberal arts major … They'll walk away with something they can use in whatever their major might be. To my knowledge, it's the first sort of serious effort at doing something like this" (Wood and Bix, 2014). The course has separate groups of students, divided into humanities, social science, and economics. It teaches basic Python programming to all three groups, and then the groups have separate sessions with lessons that focus on their own interests. According to one of the instructors, Dennis Tenen, the humanities group focuses on text analysis.[9] The students learn how to program by writing code to analyze their own text. For example, their programs search the texts for the most frequently used words. Another exercise advocated by Tenen is for students to improve their papers by removing "weasel words", vague authoritative phrases such as "popular wisdom has it" or "it is known that." This method of learning brings home the value of computing in a very personal way, and Tenen says it is very successful. In contrast to my course, Columbia's is a full semester course on programming, and students must write a lot of their own code. Furthermore, my course focuses on AI rather than text processing. However, given Tenen's success, I think that more text processing would be a great benefit to my course.

It would seem natural that the field of digital humanities (DH), which integrates electronic media with the humanities, would teach basic programming. After all, DH practitioners seek novel dynamic visualizations and exciting interactive presentations of humanities material.

[9] Personal communication, February 2015.

However, my investigation of the curricula of several universities with DH programs that are listed on the Humanist Discussion Group[10] indicates that programming is not standard in most of these curricula. Brigham Young University stands out with offerings tailored to humanities students, featuring classes in LiveCode, a programming language that hides many tedious details of traditional programming, similar in concept to Python. However, LiveCode does not have the built-in support that Python does for processing language. Some of the other schools have optional programming tracks, but the tracks are not geared towards non-STEM majors. Other schools do not mention programming in their curriculum. Since corpus linguistics, the analysis of texts, is central to DH, a programming course focused on language seems ideal for any DH program.

Computational Linguistics and NLP

Computational linguistics is another way that the humanities intersect with computer science. There are many universities world-wide that have computational linguistics/NLP courses, as indicated by a survey of the Association for Computational Linguistics.[11] However, most of these teach NLP at the graduate level. They cover a lot of statistical and mathematical theory and assume a significant background in programming. While not using any technical terms, my course lightly presents the use of a probabilistic language model derived from text corpora, as discussed in more detail below. Macquarie University in Australia has a beginning programming course in which students create a program that can hold a conversation, similar to my course. However, Macquarie's course is in Prolog, a language that is centered on predicate logic rather than a traditional programming paradigm. I believe that students are better served learning the more mainstream constructs of Python since my goal is for students to understand traditional

[10] http://tanyaclement.org/2009/11/04/digital-humanities-inflected-undergraduate-programs-2/
[11] http://www.aclweb.org/aclwiki/index.php?title=List_of_NLP/CL_courses

programmatic behavior. Closest to my project is Brandeis University's Ling:131 "Programming for Linguistics" graduate class, which teaches a good deal of NLP in Python using a book that is geared towards Python beginners, but covers a lot of ground and quickly dives deep into complex programming territory (Bird et al, 2009). Thus, the Brandeis course is not for beginning programmers, nor is the material suitable for a short course (three weeks) on the basics of programming. However, it does present some fun and relevant NLP examples, such as comparing writing styles of well-known authors or tracking the change in vocabulary of presidential speeches. It might be worthwhile to incorporate some examples from this book that could be covered in an undergraduate class for non-STEM majors.

Notwithstanding the appeal of these courses that avoid math and statistics, we cannot ignore the influence of statistical approaches on NLP. In the statistical approach, a program does not focus on the meaning of words, but rather the distribution of the words. That is, when examining a word, the program looks at the words that occur just before the word and just after it. Using distributional data, a program can compute the probability of one word being followed by another, and this basic statistical probability is very useful for many NLP applications. For example, a machine translation program might translate text from Spanish to English. This program begins by reading a collection of millions of previously manually translated Spanish-English pairs of sentences. These sentences, produced and verified by human translators, provide the distributional data that indicate which Spanish word is often translated to which English word. Of course, a lot of translated sentences are needed for the statistical method. The program also has data from perhaps billions of English-only documents that provide statistics on which English words occur around other English words. Once data are obtained, the process of translating becomes a process of computing counts and probabilities. As might be expected, the

rise of the Internet with ever expanding corpora of English and Spanish texts means that more and better statistics will continue to be available, which means translation programs will continue to improve.

Statistical approaches have been instrumental in the astounding AI and NLP successes of the last decade, as we have witnessed rapid advances in speech recognition, voice-controlled queries in applications like Apple's Siri, improvements in machine translation displayed by Google Translate, the victory of the IBM Watson program over champion human Jeopardy players, and the much anticipated introduction of driverless cars.[12] The statistical approach does not use methods that we would consider true intelligence. After all, the program is not using meaning or world knowledge to choose the best translation; nevertheless, the results are acceptable a lot of the time. The progress due to the statistical approach makes it appear that true machine intelligence is just around the corner, and several eminent AI researchers recently penned an open letter that recommends future research initiatives to ensure that these technologies not be misused or abused.[13] The letter was signed by scores of respected scientists, including science superstar Stephen Hawking. Appended to this letter was a thoughtful and detailed list of recommended research programs. Among these programs is a long-term research program to study safeguards against the dangers of creating an intelligence explosion which could possibly result in super intelligent machines which would have "a major impact on society." The letter clearly stated that whether or not such an intelligence could be created is highly speculative and controversial, but nevertheless the possibility is non-negligible over the long term.

Of course, with no time span given, practically anything is possible. The true focus of this letter is the concern for the misuse of current statistically-driven technology, not results that may

[12] For driverless vehicles, the AI involves, not collections of sentences, but rather collections of the human steering and breaking actions that occurred during countless trips along the road.
[13] http://futureoflife.org/misc/open_letter

never happen. Yet public focus swiftly turned to the long term non-negligible possibility of machines taking over humanity. For example, the Daily Mail's report: "In the long term, (AI) could have the potential to play out like a fictional dystopia in which intelligence greater than humans could begin acting against their programming." (Zolfgharifard, 2015). This view of AI helps push people's naïve belief in science fiction, where robots typically exhibit human-level conversational ability. Thus, an important part of my class, once computer programming is demystified, is to raise the curtain on the current state of AI. In AI's current state, speech recognition, machine translation, and question answering have nothing to do with understanding language in the usual sense;[14] it is all a matter of statistical probabilities and word associations. Consequently, most teaching material on computational NLP techniques involves a significant amount of probability theory and mathematics. My goal was to introduce just enough of the concepts so that students would understand the difference between deep semantic processing, a path that could lead to human-like intelligence and statistical association, a path which seems far more limited. In the former, a computer program would probably have to have data about the structure of language as well as data on linguistics theories of presupposition and explictures. Thus, this language introduction sets the stage for the subsequent course material in semantics and pragmatics as well as a critical analysis of how AI is portrayed in pop culture.

Outreach for Education in Computer Science

Experts and educators generally agree that we need more computer science education. There are not enough software professionals to satisfy the needs of high technology companies, government agencies, or even businesses in general. Software expertise is needed across the board in an ever increasing connected and information-driven world. Related to, but not the same

[14] Most researchers follow McCarthy's definition (McCarthy, 1990) of understanding a text, which means to be able to answer arbitrary questions about a text, make summaries, and make common sense inferences (Mueller, 2000).

issue, is the recognition that more focused effort is needed to interest girls in STEM Fields. Teaching girls to program is important to this study because, as cited earlier, the majority of humanities students are female. For a variety of reasons, girls shy away from STEM fields, and we continue the vicious circle of vast underrepresentation of female professionals and executives in these fields, which discourages female participation, fosters stereotypes, and leads in turn to a continuation of the same problem of lack of girls' interest (Bach, 2015).

Researchers believe we can increase young people's, especially girls', participation in computer science by making programming more of a social activity and more collaborative process (Barker, 2005). One interesting idea is Storytelling Alice, which introduces programming to middle-school girls by allowing students to make up stories and present them in 3-D animations. The students write code to manipulate the characters and display dialog. Experiments showed that girls were much more likely to be engaged in programming when they were given story-telling capabilities (Kelleher and Pausch, 2007). Along these lines, the Scratch programming environment lets kids code games and share tips (Tiku, 2014). Another example of a focus on fun animation is the Logo programming language, which lets primary school children write code to manipulate the movement of a turtle (Diehl, 2009). Raja (2014) reports children as young as fifth grade successfully programming in Logo. At the high school level, the state of Alabama recently started teaching art students about computer science through programs that write music and create interesting art designs. This Alabama initiative, supported by the National Science Foundation (NSF), reports success at enrolling young women and underrepresented minorities (Dubrow and Hendrickson, 2015). Also supported by the NSF, the CS 10K project aims to get new computer science curricula into 10,000 high schools (Dubrow, 2015). Besides

advocating Python and Scratch environments, CS 10K advocates teaching with CodeAvengers,[15] a web site that gives young students hands-on experience building web pages, building simple apps, and creating games. Clearly, all of the specialized teaching environments listed here play into the same idea of motivating students to learn coding through interesting content. For humanities majors, NLP-based content certainly seems an appropriate choice.

The above initiatives have the goal of attracting more students, especially girls, to choosing computer science as a career. However, my purpose is not to create more professional coders. Not everyone wants a career writing software, or even a career in a STEM profession, despite the recent trend that has seen negative publicity about humanities degrees and relatively large decreases in the number of humanities students with increases in the STEM fields (Levitz and Belkin, 2013). Humanities education remains fundamental to the university's providing an excellent education; humanities majors are highly sought after for their critical writing and thinking skills. However, humanities majors will be more marketable and effective at whatever job they choose if they are code-literate.

Invented Languages in the Classroom

Learning and using invented languages is rising in popularity, thanks to pop culture successes like "The Lord of the Rings" movies, "Avatar," "Star Trek" re-runs and remakes, and the HBO series "Game of Thrones" (Racoma, 2014). According to Arika Okrent, author of *In the Land of Invented Languages* (Okrent, 2009), there are several college-level courses that use invented languages to teach linguistics,[16] including ones taught at Wellesley and Swarthmore. Students at these schools use constructed languages as a vehicle to learn about phonemes, phonological rules, morphological classification, syntactic structure, language change over time, dialectal

[15] http://www.codeavengers.com/
[16] Personal correspondence, 2014.

variation, and writing systems. In these classes, students even have a final project of inventing their own language. According to Nathan Sanders at Swarthmore, his course is very popular; he is currently authoring a book on constructed languages in the classroom.[17] Clearly, these courses can go into more depth in a full semester course than the three weeks I reserved in my course for this topic, but these new courses show that invented languages are a worthwhile vehicle for teaching linguistics topics.

[17] http://sanders.phonologist.org/ as well as personal correspondence, 2015

CHAPTER 3: COURSE CONTENT

Introduction

This chapter describes the seven units of the course in detail, with class material included in the appendices. Unit 1 begins with Alan Turing's famous test for intelligence, in which a person has a "conversation" with a machine via teletype and then must decide if his or her interlocutor is a human or computer. We then discuss Turing's life and his groundbreaking ideas about machines and intelligence. Unit 2 introduces the basics of algorithms necessary for computer programming, as well as elementary coding in the Python programming language. The unit includes a small project for hands-on experience with creating a program to simulate intelligent conversation.

Having finished Unit 2 with an understanding of how computer programs work, students have gained enough programming knowledge to begin a three-unit section on how programs can intelligently handle language. Unit 3 starts this mega-section with a presentation of the fundamental challenge of AI: how to program computers to understand the language that one might find in a newspaper article or story. In this case, we use the definition of "understanding" suggested by John McCarthy (1990), whereby a program is able to answer simple commonsense questions about what it reads. We study various elements of semantics and pragmatics and how they might be handled by an NLP program, but we find that common sense, or "human experience," is a key missing ingredient that would make deep processing possible. Nevertheless, state-of-the-art NLP programs such as Google Translate and Siri appear to the user to be very intelligent even though they are not programmed with a commonsense understanding of language and context. How are they able to do this? Unit 4 explores this question, and we briefly look at how these programs use statistics and pattern matching to achieve impressive

results. But once students see how these programs work, they understand that statistical techniques will probably not be enough to achieve the deeper understanding that would be possible if computers had commonsense data. Thus, Unit 5 presents the problem of getting this type of data into a computer. It covers several projects that have attempted to collect commonsense data for AI and discusses why none has yet succeeded.

At this point, students understand the huge obstacles involved with NLP. Armed with this knowledge, Unit 6 looks at how AI is presented in pop culture, examining science fiction film and television classics *2001, A Space Odyssey* (1968) and *Star Trek: The Next Generation* (1987-1994). The underlying theme in this unit is the illogic of supposedly logical behavior in AI-related films and television. That is, the artificial agents are sophisticated enough to handle the subtleties of conversational explictures and the complex inference that would be expected from a person with considerable life experience. Yet comparatively simple tasks, such as understanding puns or recognizing human subterfuge, are beyond the agents' capabilities.

After the unit on AI in pop culture, we shift gears into the topic of invented languages. Unit 7 explores language topics according to students' interests, as the students themselves do their own research and make presentations in class on the phonology, syntax, and socio-linguistic issues of invented languages.

What follows is a detailed description of each course section and results of teaching the course on a Tuesday/Thursday schedule during the fall semester of 2014. Class met 80 minutes per session.

Unit 1: The Turing Test: Language as a measure of intelligence

Unit 1 (2 class sessions) introduces students to Alan Turing, his work on code breaking during World War II, and his groundbreaking paper on artificial intelligence, "Computing Machinery and Intelligence" (1950). I chose to start the course with Turing because he is an excellent bridge between humanities and computer science. A consummate scientist and theoretical mathematician, he was nevertheless interested in language, and his paper on machine intelligence is accessible to non-STEM audiences. In it, he poses the question, "Can Machines Think?" and presents a thought experiment to help answer this question. This experiment, now called the Turing Test, is still considered the litmus test for machine intelligence.

In his paper, Turing presents a parlor game, "The Imitation Game," played with a man and a woman, both hidden from view, and a third person–the interrogator. The goal of the game is for the interrogator to ask the other two players questions to try to determine which one is the woman. Communication is via slips of paper so that the physical differences between the man and woman cannot be detected. To make the game more interesting, the man is allowed to be tricky and pretend to be the woman, while the woman is instructed to answer all questions honestly. Turing suspects that in some cases the interrogator will be fooled and in some cases not. Now, says Turing, suppose we substitute a computer for the man. If the computer can fool the interrogator as much as the tricky man, we should consider the computer to be intelligent.

When I taught the course, on the first day I briefly introduced Turing's Imitation Game, and then we played it. I had brought in a screen that could hide two people in front of the class's computer. The two hidden people typed on the keyboard in response to oral questions posed by the rest of the class. This was an excellent first-day ice-breaker, the class seemed to enjoy it a lot, and it paved the way for more serious discussions to follow.

The unit has discussions on Turing's achievements as well as Turing, the man. It starts with Turing's role in breaking the code of the Nazi Enigma Machine during World War II. The Enigma had several possible mechanical configurations that allow it to create millions of coding possibilities, and it was changed regularly; thus, traditional code-breaking methods with paper and pencil would simply be too slow, and the Germans considered the system unbreakable. To solve the problem, Turing designed a code-breaking machine, a specialized computer that could cycle through the vast majority of code settings and rule out the impossible ones very quickly. The code settings that remained after this process were few enough so that they could be solved by hand. Crucial to using Turing's machine was an inherent attribute of human language that is still necessary today for all programs today that "understand" language: people communicate in predictable patterns. In the case of the people using the Enigma, the Nazi commanders, many of them opened a communication with a weather report (whose words could be guessed by the code breakers) or a standard phrase such as "All Clear." In the case of a modern program that answers questions on a smart phone, programmers also look for various predictable patterns and key phrases.

To further engage humanities students in Turing's work, the unit includes a discussion of the movie adapted from the play about Turing's life, "Breaking the Code" (1997).[18] In one particularly interesting scene, Turing's character discusses science with his fiancé and then reveals that he is a homosexual. Turing was later prosecuted for homosexuality in 1952 despite being considered a war hero. Thus, the play's title, "Breaking the Code" includes the double meaning of Turing breaking the Enigma code as well as society's heteronormative code.

[18] The 2014 Movie *The Imitation Ga*me, also based on Turing's life, was released at the end of December, too late for this class. However, it would be an important addition to future iterations of this course.

Turing's hidden sexual orientation also adds significance to Turing's imitation game, where a man is pretending to be a woman.

After the material on Turing's personal life, the rest of this short unit covers his 1950 paper and his thesis about whether machines can think. Turing's ideas of machine intelligence are based on *functionalism*–outward behavior rather than what is believed about internal mental processes. Turing's idea is that people who don't think the machine is thinking have a solipsistic point of view, which is the view that the only way to believe a person is thinking is to actually *be* that person. According to Turing, if you have a solipsistic point of view, you cannot in principle believe that anyone else is thinking. Of course, Turing argues, this is ridiculous, so you are forced to accept that people are thinking by how they interact with you, and thus you should give a machine the same consideration. That is, you should judge intelligence based purely on the ability to have a conversation; if the machine fools you in the imitation game as well as a human, the machine should be considered intelligent.[19] Turing refutes many hypothetical arguments against his thesis: the religious argument that machines cannot have a soul, the argument that machines cannot be creative, the argument that machines cannot have self-awareness, and many others, including the bizarre argument that a machine would not be able to compete in the imitation game with a man who has extra sensory perception (ESP). In all but the ESP argument, Turing maintains the functionalist position that the machine should be judged on how it behaves and not what it has going on inside. Machines can certainly appear to be creative, self-aware, dishonest, and all the other traits that humans have. As far as ESP, Turing concedes that the

[19] AI workers have devised conversational strategies that require very little actual language understanding to simulate superficial conversations. Therefore, simply the ability to hold a superficial conversation does seem to be an adequate test for intelligence. McCarthy's proposal that the program be able to answer arbitrary questions about a story (discussed in Chapter 3, Unit 3) seems to be a better indicator of language intelligence.

game should be housed in a "telepathy-proof room" so that a person with ESP does not have an advantage![20]

It is important to note that the Turing test is a thought experiment. Machines are not even remotely close to simulating understanding of language, despite our imaginations and what is presented in science fiction. A program called Eugene that behaves like a 13-year-old Ukrainian boy has recently won a version of the Turing by fooling 30% of the judges (Amlen, 2014). However, when we look at the actual language interactions that this program is reported to have, it does not seem that this version of a Turing test goes far enough to test intelligence. Eugene's responses display neither context awareness nor other factors that indicate intelligence. Eugene is a "chatbot," a program that allows a person to type sentences and get responses back. Cleverbot,[21] is another example of a chatbot. Conveniently, Cleverbot is online and the class experimented with questions and responses. In experimentation, it becomes clear that a chatbot looks for key words and responds from a fixed set of canned answers. And when a chatbot cannot find a suitable response, it typically gives a generic non-sequitur response that changes the subject.

The material for this unit is in Appendix 3. It consists of the transcripts from the first day's activity and a homework exercise that tests students' understanding of the material.

Unit 2: Computational Thinking: How computer programs work

After learning about Turing and chatbots in Unit 1, students are now ready to learn the basics of programming. Unit 2 (5 class sessions) introduces algorithms, shows what a program looks like, and presents the programming constructs of input, output, variables, assignment statements,

[20] Turing might have been writing tongue-in-cheek about ESP, although he was probably serious (Oppy and Dowe, 2011).
[21] www.cleverbot.com

if-statements, while-loops, functions, and list manipulation, all in the Python programming language, which is a popular language for natural language processing.

An everyday example of algorithmic thinking is what happens when we give someone directions. For example, when asked for detailed directions to the campus library, students may come up with something like the following:

1. Leave the classroom.
2. Turn left.
3. Go the end of the hall.
4. Walk out the door.
5. Walk along the sidewalk until you see the library

A computer could not follow these instructions because computers do not understand commands like "leave the classroom" or "walk out the door." I ask students to imagine that there is a robot that can obey the following set of instructions: Get up, Turn Left, Turn Right, and Take Step. Using these commands, students attempt several times to write instructions to get to the library. After each attempt, I pretend to be the robot and follow the instructions to the letter. For examples, if there are desks in my way, I run into them. If there is a closed door, I walk into it. This exercise helps students see the difficulty of writing precise commands that a machine could follow. Students find that that they have to count the exact number of steps required before each turn, and there need to be new commands for grasping a door knob, turning it, and pulling the door. By the end of this exercise, students comprehend how unintelligent computers really are and how algorithms must unambiguously lay out every detail.

Next we started to learn about programming in Python. Through a series small programs of ever increasing complexity, students learn the basics of programming a conversational agent, a

chatbot like Eugene or Cleverbot discussed in the previous section. Appendix 4 shows the programs used for this unit. Students do not do any actual programming on their own. The goal here is to learn how a program works and to understand what programs can do, not to become a programmer. For students to write and execute their own programs, I would have had to teach them how to install Python, edit Python files, and deal with a host of enigmatic error messages. The pain and suffering entailed in such an enterprise would be worth it only to those seriously interested in learning to code, and it would take a full semester. Nevertheless, as discussed below, this unit does give students some practice writing code. These 12 graduated programs are named P01_hello_world, P02_sounds, P03_chat, etc. through to the 12th program called Eliza_Students, as detailed next.

1. P01_hello_world

   The first program in the lesson is called p01_hello_world.py. Typical of any beginning program class, it simply prints "Hello World" to the computer screen. It introduces students to what a computer program is as well as the concept of "output." In this case, the text on the screen is the output.

2. P02_sounds

   The second program is p02_sounds.py. It expands the definition of output to include not only writing to the screen, but making sounds–beeps and a bizarre sound recorded to a file. The idea is to demystify how programs can produce different types of output.

3. P03_chat

   The third program is p03_chat.py. It is the first step in creating a chatbot. It prompts the user for a name and then says “Hello” with the name. The goal is to show how input and output combine to form the illusion of a conversation.

4. P04_chat

   In the fourth program, we discuss variables and how they should have meaningful names to make it easier for other people to understand our programs. To illustrate this idea, P04_chat is exactly the same as P03_chat, except that the meaningless variable “x” is replaced with the more descriptive variable “name” to indicate that the variable represents the name of the user.

5. P05_chat

   The fifth program contains the first example of an if-statement, a construct that acts like a fork in the road. An if-statement causes the program to branch one way or the other depending on the value of a variable. In the example below, the program asks the user to type his or her name. Then depending on what the user types, the program substitutes a nickname.

```
print "What is your name?"
name = raw_input()

if name == "Thomas":
    name = "Tommy”

if name == “Elizabeth”:
    name = "Lizzie”

if name == “Barack”:
    name = "Mr. President”
```

With this type of construct, the program can print out a user's nickname during the conversation.

6. P06_chat

   The sixth program has a more elaborate version of an if-statement, called the if-elif statement. This is a simpler and more efficient way to have multiple if-statements.

7. P07_chat

   The seventh program introduces the while-loop statement to allow a continuous back-and-forth conversation. It also shows manipulation of strings of characters. First, the program checks if the user has typed in a string with the words "I want to". If so, the program replaces any string "you" with "me" and replaces any string "my" with "your". Finally, the prints the resulting string, headed by "Do you really want to". For example, if the user types "I want to meet your teacher", the program would print out 'Do you really want to meet my teacher?".

   Besides this very specific behavior for the input 'I want to', the program has some more general patterns it responds to. Upon receiving any input with the word "yes", the program will print "How can you be so positive?". Upon receiving any input with a question mark at the end, the program will print "I'm not in the mood to answer your questions".

   With this program, students have learned the basics of how programs search for key words and phrases and print canned output that appears to be relevant to the input. If students understand this basic structure, they will comprehend that any program that appears to be intelligent simply has a lot of search patterns and interesting canned responses that the programmer has provided.

8. P08_chat

   The previous program has a severe limitation: it always gives the same response to the same input. This repetitive behavior would never fool anyone. To address this problem, the eighth program creates long lists of different responses for the same input. Furthermore, the program shows how the *random* function chooses a random item from the list, causing the output to appear to be unpredictable. Thus, students see how programs exhibit novelty and unpredictability, two hallmarks of what humans perceive to be intelligent behavior.

9. P09_lists

   The ninth program is a tiny tutorial in list manipulation. Lists are fundamental to programming because they allow programs to keep track of more than one item. This program shows how programs can append items to a list and check to see if an item is in the list.

10. P10_chat

    The tenth program returns to the chat example. It demonstrates how to solve a problem in the eighth program in which the random function, because it is random, chooses a response from the list that has previously been chosen. This produces repetitive behavior, something we need to avoid. Students see that solving this problem is not trivial. To keep track of which responses have already been used, the program puts all used responses in a list. Before printing a new response, the program checks to see if the list already contains that same response. If so, the program selects a different response (using a while-loop) until it finds a fresh response.

11. P11_chat

    The previous program is difficult to read because of the complexity of searching for unique responses. The eleventh program shows how code can be divided into functions to make the complexity more manageable and readable. In this case, the part of the code that searches for a fresh response is now located in a separate function called get_response. With this structure, any part of the program can call this function to get a unique response.

12. Eliza_students

    The ELIZA program (Weizenbaum , 1966) was an early example of a chatbot. It was programmed to imitate a psychotherapist. Since psychotherapists are known to be interested in the family life of their patients, one of the famous traits of the ELIZA program is that it looks for key words such as “mother” or “father.” When it sees such as word, it might respond with a question such as “Tell me about your mother” or “Do you respect your father?” Thus, if a person types, “I saw my mother today. We had lunch and went shopping.” The program would simply key-in on “mother” and respond with, “Tell me about your mother.” A version of the ELIZA program is available for download.[22] The program makes use of regular expressions to search for patterns in the input. The following is a regular expression that matches any input in the form: “I feel X”

```
[r'I feel (.*)',
[  "Good, tell me more about these feelings.",
"Do you often feel %1?",
"When do you usually feel %1?",
"When you feel %1, what do you do?"]],
```

[22] http://www.nltk.org/api/nltk.chat.html

> For example, if the user types "I feel like going to a movie", the program would respond with one of four sentences: "Good, tell me more about these feelings," "Do you often feel like going to a movie?", "When do you usually feel like going to a movie?", and "When you feel like going to a movie, what do you do?" Using regular expressions is an excellent way to teach students about pattern matching. In this unit, students learn the structure and syntax of regular expressions, and then they are assigned a small project to create their own expressions and responses.

When I taught the course in fall 2014, I showed the class some interesting work from poet and LSU MFA student Laura Theobald. Theobald uses Cleverbot as part of her poetic process, and she has written poems consisting of what she types and Cleverbot's responses.[23] To motivate students to be creative with their own expressions for their projects, I invited Theobald to come to class and use our home-grown chatbot. I combined all of the regular expressions that the students created into one chatbot, and we watched while Theobald posed questions and interacted with our program. Unfortunately for the class, Theobald did not hit any of the fun regular expressions that the students had created. Watching someone type in random inputs to a chatbot, the students quickly learned how difficult it is to simulate an intelligent conversation. Most of the students' expressions were overly specific. For example, they looked for a particular phrase such as "What is your favorite X?" so that they could provide a witty response like "My favorite X! What is this a first date?" But after seeing Theobald's interaction with the chatbot, the students realized that it is very unlikely that someone would type in the input that would match a specific expression unless there are perhaps tens of thousands of such expressions. Thus, writing a good chatbot requires a prodigious amount of work.

[23] See Theobald's Cleverbot poems at http://cleverbotmyonlyfriend.tumblr.com/

Unit 3: Computational Semantics: Why language is difficult for computers to process

After completing Unit 2, students understand the basics of how programmers create programs to simulate conversation. The goal of Unit 3 (3 class sessions) is for students to understand the enormous difficulties involved in programming computers to understand language in general. The unit first presents the problem of understanding newspaper articles, as formulated by McCarthy in 1976 (McCarthy 1990, p.70) and reinterpreted by Mueller (2000). Using an article about a furniture store robbery as an example, McCarthy shows that any intelligent reader should be able to answer simple questions about the reported event. McCarthy asks how a computer could answer these questions. Many years later, Mueller(2000) makes the case that a computer would have to be programmed with commonsense knowledge about robberies, elevators, the size of the human body, and much more to hope to answer these types of questions. Writing around the same time as McCarthy, Charniak (1976) also attacks the language problem, but he proposes to start with the more straightforward language of children's stories in order to simplify the task. However, even Charniak's level of text has sophisticated language structure, and so this unit aims to analyze the simplest possible children's stories, those for audiences of 3-6 years in age. To provide a framework of analysis, the unit discusses eight challenges to natural language processing, most of which are familiar topics in semantics and pragmatics. They are 1) Word Sense Disambiguation; 2) Entailment; 3) Structural Ambiguity; 4) Pronoun Resolution; 5) Presupposition; 6) Bridging Reference; 7) Conversational Implicature; and 8) Answering Common Sense Questions. Through the lens of these eight phenomena, we analyze a simple children's story to see what a programmer would have to do to simulate human-level understanding of the text. By the end of this unit, students have the linguistic tools necessary to understand the AI programmer's task of natural language processing (NLP).

Appendix 5 has detailed analyses of several sentences from a children's story that students can use to model their own analysis. The appendix also has exercises to test the students' understanding of the eight challenges of NLP. I taught this unit as a lecture with the following material as notes available to the students.

Introduction

In an article on computational semantics, Erik Mueller (2000) refers to a problem first posed by John McCarthy in 1976 (McCarthy 1990, p.70). McCarthy quotes a story from a newspaper article and asks whether a computer program could be written to understand such a story:

> A 61-year old furniture salesman was pushed down the shaft of a freight elevator yesterday in his downtown Brooklyn store by two robbers while a third attempted to crush him with the elevator car because they were dissatisfied with the $1,200 they had forced him to give them. The buffer springs at the bottom of the shaft prevented the car from crushing the salesman, John J. Hug, after he was pushed from the first floor to the basement. The car stopped about 12 inches above him as he flattened himself at the bottom of the pit.

Mueller points out that the problems raised by McCarthy remain unsolved. No program can understand the above story at a level anywhere near that of a human reader because this type of understanding probably requires common sense, something that is still out of reach from current technology. Only a limited understanding is possible in the current state-of-the-art NLP. Mueller cites some progress in NLP, especially regarding the simpler problem of *Information Extraction*. Given a particular class of events such as terrorist incidents, programs can be built that extract basic information such as the type of incident, perpetrators, and targets. For example, a program could process the above story and fill a template with information such as the following:

- Incident: Location United States: Brooklyn (CITY)
- Incident: Type Robbery
- Perpetrator: Individual ID "two robbers"
- Human Target: Name "John J. Hug"

- Human Target: Description "61-year old furniture salesman"

Mueller also discusses progress in another difficult NLP issue, Word Sense Disambiguation. (WSD). Most words have many different possible meanings, and it is a challenge for a computer to determine which meaning to use for a given context. At the time of Mueller's 2000 article, many researchers believed that programmers would need a lot of commonsense data and context awareness to make good choices to disambiguate words. The robbery victim was pushed down an elevator shaft, but a computer could interpret the word *push* to mean to sell drugs or to advertise a product, for example. The word *down* has many meanings, from down feathers to downing a drink. However, in the years since Mueller's article, a great deal of progress has been made in WSD without using commonsense data, as can be seen with impressive performances of systems that handle Machine Translation. Using statistical techniques, Google Translate[24] does a very good job when translating the newspaper article above from English to Spanish in April 2015. However, there was still one significant WSD error in Google's translation due to the word sense ambiguity of *shaft*, which can mean, among other things, an axle of a vehicle or a narrow passageway. The Spanish translation said that a spring at the bottom of the axle stopped the elevator, rather than a spring at the bottom of the shaft (of the elevator). Nevertheless, as more translated data become available, translation programs should make fewer WSD errors, although other translation challenges such as pronoun resolution, as discussed in the next unit, still remain difficult due to their apparent dependence on common sense and context.

Mueller points out that McCarthy's concern with language is much deeper than information extraction and WSD. McCarthy proposed that a computer should be able to demonstrate its understanding of language by answering questions about the above story such as:

- Who was in the store when the events began?

[24] translate.google.com

- Who was in the store during the attempt to kill Mr. Hug?
- Who had the money at the end?
- What would have happened if Mr. Hug had not flattened himself at the bottom of the pit?
- Did Mr. Hug want to be crushed?
- Did the robbers tell Mr. Hug their names?
- Did Mr. Hug like the robbers, and did they like him?
- What would have happened if Mr. Hug had tried to run away?
- What can Mr. Hug do to avoid this in the future?
- Did Mr. Hug know he was going to be robbed?
- Does he know that he was robbed?
- How did the robber try to crush him with the car?
- How long did the events take?
- What crimes were committed?

...
(Excerpted from McCarthy, 1990, pp. 70-71)

Mueller asserts that commonsense data must be loaded into the computer in order for it to answer these questions.

Charniak was a graduate student at the time McCarthy was noting the difficulties in understanding text. Aiming to simplify the problem, Charniak (1972) focused on the commonsense data that would be needed to understand children's stories. Along the same lines as McCarthy but changing the example to be a text for young readers, Charniak (1976) presented the following story and discussed how a computer would be able to understand it.

> One Saturday Jack decided he wanted some cereal for breakfast. However he had to go to the supermarket, where, after finding the shelf where the milk was, he payed for it and left. When he got home he found that the milk was sour. "Why do these things always happen to me," thought Jack.
>
> Questions:
> 1. Why did Jack go to the supermarket?
> 2. What did Jack pay for?
> 3. By the last line had Jack finished his cereal?
> 4. What was Jack complaining about?
>
> Common Sense:
> 1. Normally, cereal is eaten with milk.
> 2. Supermarkets primarily sell food.
> 3. Sour milk tastes bad.
> 4. Except under special circumstances one quickly stops eating bad tasting food.

Pronoun resolution and deixis:
1. What does “it” refer to?
2. What do “these things” refer to?

Although not as difficult to understand as a typical newspaper article, the story about Jack and the spoiled milk does have relatively complex sentence structure and somewhat subtle inferences about why Jack would be frustrated. Going even further than Charniak to simplify the problem, a group of AI researchers, including McCarthy and Mueller, proposed to analyze text for extremely young audiences to see what data a computer program would need to reach human-level comprehension in this very limited domain (McCarthy et al., 2002). In these stories, there are very few subtleties of expression that could confuse a very young reader. Nevertheless, there are still a vast amount of implicit life experiences that the reader is assumed to have. This unit takes up the challenge of those researchers as we analyze what sort of information must be in a computer program in order to answer arbitrary questions about a seemingly simple story. As part of this analysis, this unit briefly introduces different types of NLP problems.

The next sections briefly describe eight important problems of NLP and ask what information a computer program would need in order to understand language at a deep semantic level—the level proposed by McCarthy. This list is ordered below by the amount of commonsense data that a computer is presumed to need in order to be able to handle the linguistic phenomenon. After this introduction, this unit analyzes several sentences from a simple story in terms of these eight challenges.

Eight Challenges for NLP

1. Word Sense Disambiguation

Almost every word has multiple meanings. How does a computer know which meaning to choose? For example, consider the sentence: “The scientists look for a good solution.”

Assuming we know the correct part of speech, the lexical database WordNet (Fellbaum, 1998) has many definitions for the verb *look* and the noun *solution*. WordNet's online data for these words is shown below in Figure 1. [25]

Verb

- S: (v) **look** (perceive with attention; direct one's gaze towards) *"She looked over the expanse of land"; "Look at your child!"; "Look--a deer in the backyard!"*
- S: (v) **look**, appear, seem (give a certain impression or have a certain outward aspect)*"She seems to be sleeping"; "This appears to be a very difficult problem"; "This project looks fishy"; "They appeared like people who had not eaten or slept for a long time"*
- S: (v) **look** (have a certain outward or facial expression) *"How does she look?"; "The child looks unhappy"; "She looked pale after the surgery"*
- S: (v) search, **look** (search or seek) *"We looked all day and finally found the child in the forest"; "Look elsewhere for the perfect gift!"*
- S: (v) front, **look**, face (be oriented in a certain direction, often with respect to another reference point; be opposite to) *"The house looks north"; "My backyard look onto the pond"; "The building faces the park"*
- S: (v) attend, take care, **look**, see (take charge of or deal with) *"Could you see about lunch?"; "I must attend to this matter"; "She took care of this business"*
- S: (v) **look** (convey by one's expression) *"She looked her devotion to me"*
- S: (v) expect, **look**, await, wait (look forward to the probable occurrence of) *"We were expecting a visit from our relatives"; "She is looking to a promotion"; "he is waiting to be drafted"*
- S: (v) **look** (accord in appearance with) *"You don't look your age!"*
- S: (v) count, bet, depend, swear, rely, bank, **look**, calculate, reckon (have faith or confidence in) *"you can count on me to help you any time"; "Look to your friends for support"; "You can bet on that!"; "Depend on your family in times of crisis"*

Noun

- *S: (n)* ***solution*** *(a homogeneous mixture of two or more substances; frequently (but not necessarily) a liquid solution) "he used a solution of peroxide and water"*
- *S: (n)* ***solution****, answer, result, resolution, solvent (a statement that solves a problem or explains how to solve the problem) "they were trying to find a peaceful solution"; "the answers were in the back of the book"; "he computed the result to four decimal places"*
- *S: (n)* ***solution*** *(a method for solving a problem) "the easy solution is to look it up in the handbook"*
- *S: (n)* ***solution****, root (the set of values that give a true statement when substituted into an equation)*
- *S: (n)* ***solution*** *(the successful action of solving a problem) "the solution took three hours"*

Figure 1: WordNet Output for the verb *look* and the noun *solution*

[25] http://wordnetweb.princeton.edu/perl/webwn, May 1, 2015

Given these various meanings, a computer could translate "The scientists look for a good solution" to "The scientists perceive for a good a homogeneous mixture of two or more substances." Assuming the program has access to all of the information in WordNet, what sort of information would it need to choose the appropriate word senses so that it could understand the sentence? It turns out, as will be discussed in the next section, that often WSD can be accomplished statistically by looking at the surrounding words, without using any context awareness. For example, when *look* and *for* occur together, a program could simply assume that it means *search for* without knowing anything about the rest of the context. And when *look for* and *solution* occur together, the program could assume the appropriate sense of *solution* is *problem*.

For another widely used example in AI, take the sentence, "The baby picked up the pen." Using statistical probabilities, a computer would guess in this case that *pen* means *playpen* because this sense of *pen* is most often associated with the word *baby*. However, a commonsense understanding of the strength of babies and the weight of playpens would cause a human to favor the *writing implement* sense of *pen* in this particular sentence. Thus, common sense is still needed for many WSD problems.

2. Entailment

Entailment is the relationship between two sentences where the truth of one requires the truth of the other. In the sentences below, if the first sentence is true, the second sentence must also be true.

- Sue is walking slowly → Sue is walking
- Sue killed Sam → Sam is dead
- Sue stabbed Sam → Sue used something relatively sharp, Sue touched or pierced Sam

In general, semantic entailment of verbs is more easily handled by a computer than other aspects of NLP because it can often (but certainly not always) be determined from just the meaning of the verb and the syntactic form of the sentence. However, it is still difficult to get this information since there are no standard resources of verbs and their entailments.

3. Structural Ambiguity

A computer parses a sentence to determine its syntactic structure. During the parsing process, the computer determines the part of speech of every word along with its relationship to the rest of the words. But there are usually many possible resulting structures, and thus a sentence's meaning may be ambiguous.

Example 1 (from Mueller, 2000):

- The waiter brought me spaghetti with red sauce and wine
    - Did the spaghetti have wine on it?

Example 2:

- Sue watched the elephants in her pajamas (borrowed loosely from Groucho Marx)
    - Were the elephants wearing Sue's pajamas?

It usually requires commonsense knowledge to determine which structure is most likely.

4. Pronoun Resolution

Natural language is full of pronouns or other referential expressions. Take the following extreme example, with the pronouns underlined

> Alex saw the puppy in the window. She tapped on it to get its attention, but it wasn't loud enough. Oh well, it just wasn't meant to be.

- Did Alex tap on the puppy?
- Was the puppy not loud enough?
- Was the window just not meant to be?

How does a computer know what word a pronoun refers to? Sometimes it is easy. For example, to find the reference to *it*, a program could search the text for the most recently mentioned object or non-human. If it is *he*, the program could search for the most recent masculine name. There are linguistic theories such as Centering Theory (Brennan et al., 1987) that, for example, prioritizes subjects over objects, and these solutions often guess the correct reference without common sense. But there are many cases that are more subtle, as in the example above. Consider now the two pairs of examples below. In each case, the first sentence is the same, so the pronoun reference must depend on something in the second sentence. There are no simple theories to account for this difference; it may depend on commonsense knowledge about asking for help.

- Sam asked Jim for help. He said, "Sure." (*He* refers to Jim)
- Sam asked Jim for help. He needed to finish the job by 6PM. (*He* refers to Sam)

5. Presupposition

A presupposition is implicit background information that must be true in order for the sentence to make sense. For example, the sentence "Mary no longer plays the piano" does not make sense if Mary never played piano. It implicitly assumes that Mary used to play piano. Crucially, the implicit information in a presupposition must still hold even if the sentence is negated. Whether it is true or not true that Mary no longer plays the piano, we must still assume that Mary played piano in the past. The phrase "no longer" is called the trigger of the presupposition. That is, in any sentence of the form X no longer Y, the presupposition is that X used to do Y.

Levinson (1983) contains a list of some presupposition triggers,[26] and we discuss them in class. For example, the list includes:

- Definite descriptions, such as John saw/didn/t see **the man with two heads.**

[26] Students can access this list at http://eecoppock.info/Pragmatics/Papers/presupposition-triggers.pdf

- Factive verbs, such as Martha **regrets**/**doesn't regret** drinking John's home brew.
- Implicative verbs, such as John **managed/didn't manage** to open the door.

In the first sentence, "the man with two heads" is a definite description of something in the discourse, and this discourse item must exist in order for the sentence to make sense. In the second sentence, Martha is assumed to have drunk John's home brew. In the third sentence, the presupposition is that John tried to open the door. That is, the trigger "X managed Y" presupposes "X tried to do Y."

Many presuppositions can be done without much common sense, based solely on the patterns of the triggers. However, there are many cases where common sense is needed. For example, consider two well-known examples of presuppositions:

- She cried **before** she finished her thesis
- She died **before** she finished her thesis

In the first case, *before* is a trigger for the presupposition "she finished her thesis." But the same trigger does not work in a context with *died* because if a person dies before doing X, the person did not do X.

6. Bridging Reference

A bridging reference usually involves a definite description that depends on another concept in the same sentence.

- I was riding my bike when <u>the pedal</u> broke.
    - What pedal are we talking about? To answer this, we must know that bikes have pedals, so the pedal refers to the pedal of the bike previously mentioned.
- I drove my car to the movies but <u>the tickets</u> were sold out
    - Where were all the tickets?

    - We know that you need a ticket to go to a movie, and there are a limited number of tickets. So the answer is that the tickets were bought by other movie-goers. How could a computer know this information?
- I went to Mount Rainier and was moved by the beauty.
    - Was there somebody beautiful at Mount Rainier who moved me out of the way?
    - No. We know that mountains are beautiful, and so they have the abstract quality of beauty.

7. Conversational Implicature

A (conversational) implicature is the extra information, not explicitly stated, that humans understand to be true, but computers would not, unless the computer understands many different rules about human behavior. For example, if Max asks Mary to go to a movie, and Mary replies "I have to wash my hair," the implicature is that Mary refuses Max's offer. We make this interpretation because we assume that Mary is cooperating in the conversation, so her reply to Max was somehow relevant to his offer. Using commonsense inference, we understand that washing one's hair takes some time, and thus one will be busy during the process of washing one's hair. Furthermore, being busy means that Mary is unable to go to a movie. Of course, we also understand that Mary is not very motivated to go with Max because normally hair washing is not a pressing appointment.

Implicatures are different from presuppositions in that they can be cancelled. Mary could say, "I have to wash my hair … but I can still go." Grice developed the theory of implicatures and explained them in terms four maxims of cooperation (Reimer, 2010). More details of Grice's theory and additional examples are found in Appendix 5.

Programing a computer to understand conversational implicatures is one of the most difficult language tasks because there does not seem to be any particular pattern that the programmer can look for. Each implicature requires a great deal of commonsense data and the ability to find what is relevant in a given situation.

8. Answering Common Sense Questions

The AI critic and philosopher Hubert Dreyfus takes the idea of McCarthy's arbitrary questions to the extreme. Dreyfus says that if a computer could not answer silly questions about a story, then it cannot possibly be considered to have understood it (Dreyfus, 1992). Take as example the following examples of statements, questions, and comments about how to program a computer to give a reasonable answer.

- Sue lost her purse. She looked everywhere.
    - Q: Did Sue look in her mouth? Did she look on Mars?
      A: No. Sue only looked where it would be reasonable to look. How does a computer know where it would be reasonable?

- Sue's pet cat died. So she didn't feel like going to the party.
    - Q: Why didn't Sue feel like going to the party?
      A: Sue was sad. How does the computer know when people get sad and what they do when they are sad?

- President Obama was in Louisiana on Wednesday.
    - Q: Was Obama's right arm in Louisiana on Wednesday? Was his bowling ball in Louisiana on Wednesday? Was his assistant in Louisiana on Wednesday?
      A: Yes, Probably Not, Probably Yes. How does a computer know what things are with a person when the person is at a location?

None of these questions would be difficult for a typical reader, yet a computer must be programmed with commonsense data and the ability to make inferences in order to give an intelligent answer.

Analysis of a Children's Story

After discussing the eight challenging aspects of NLP for computational processing, the rest of this unit consists of analyzing a very simple children's story in terms of each of these eight aspects.[27] The story is *Angelina and the Butterfly* by Katherine Holabird (2006), a story for children 3-6 years old.

Appendix 5 contains the analyses presented in class of the first five sentences of the story in the section entitled "Analyses of Sentences." At the end of this unit, students are assigned a small project to do their own analysis of a different sentence from the story. Appendix 5 also has homework exercises to test student understanding of the material in this unit. After doing this project, students understand much better the magnitude of the problem of NLP and narrative. To appreciate the magnitude of the world knowledge the typical reader brings to bear when reading text, I believe it is necessary to analyze the text in fine detail. The linguistic tools learned in this unit provide a framework for this analysis. Students go down the check list and arrive at a sophisticated understanding of the challenges that the NLP programmer must overcome to achieve human-level understanding of the simplest of children's stories.

Unit 4: Statistical Natural Language Processing: How Siri and Google work

The previous units showed students how difficult it is for computers to understand natural language. Nevertheless, programs like Siri appear to understand speech and are able to answer questions. Google appears to know what we are going to ask even before we finish typing. And automatic translation programs such as Google Translate often do a pretty good job of translation. Add to that the formidable achievement of IBM's Watson program beating champion

[27] There were not any conversational implicatures found in this analysis, and there are very few to be found in the entire children's story. Presumably, this is because children are not expected to understand the subtleties of indirect communication. It would be interesting to analyze more stories for small children to see if this pattern holds.

players at Jeopardy and the much anticipated driverless cars, and it may seem as if computers could be on the verge of true intelligence. The goal of Unit 4 (1 class session) is to demystify these programs and show that, in fact, there has been very little progress towards creating machines with the type of general, context-aware intelligence that is depicted in science fiction. Rather, the current state-of-the-art NLP programs use statistical techniques to look for reoccurring patterns in a manner that requires lots of data and number crunching, but very little contextual knowledge. This one-day unit describes, using very little math, the general idea behind these mathematically sophisticated techniques. Appendix 6 contains exercises for this unit.

Unit 4 begins with an arresting online demonstration of Siri[28] that makes it look like Siri understands speech. Next, the unit outlines the fundamental problem of speech recognition: given a series of phonemes, how does a program choose which words belong to the phonemes. During class, I demonstrate the problem by speaking a series of foreign names. Since the names are unfamiliar, students are baffled by what I say, and they cannot even say where the word boundaries are. But when I tell them I am speaking a list of names, and I display a list of the possible names that I am saying, the students can figure out what I say. We discuss the fact that computers do the same thing. If you give a program a list of possible words, and the possible phonemes that make up those words, the program can guess which words are spoken by a process of elimination. So, it stands to reason, if you give a computer a list of all possible English words and their phonemes, it could recognize the words of an English speaker.

But this simple-minded process of elimination has a problem. Many words have similar phonemes, and some even have the exact same sounds. How does a computer choose which words are spoken among the many possible combinations of English words that could match the

[28] http://www.engadget.com/2011/10/05/iphone-4s-what-can-you-say-to-siri/

phonemes? As an example, consider the following three sentences, all of which contain valid English words:

1. I want a computer to recognize speech
2. I want a computer to wreck a nice beach
3. Eye wont eh come pewter two wreck an eyes peach

The answer lies in probability. Even without doing any math computations, students understand that it is much more probable that a person uttered sentence (1) than the other two sentences. Then we discuss how to write a program to guess which words are the most probable using corpus linguistics. In corpus linguistics, there is a huge body of text that is representative of acceptable speech. For example, the Penn Tree Bank (Marcus et al., 1993) is a corpus that contains thousands of newspaper articles and books as well as transcriptions of conversations. With such corpora, a computer can count each time a word is used and determine which words are the most frequent in normal speech. It can also determine which words are most likely to precede other words. For example, the word "Orleans" is most often preceded by the word "New." *Bigrams* are pairs of words in the corpora, and *trigrams* are triples of words. A computer can look at the frequency of every possible bigram and trigram in the corpora, and it uses this information to guess which words are followed by other words.

Although most of the statistical data are taken from corpora of written text, the computed frequencies also apply to speech. Thus, with statistics on word frequencies as well as the probabilities of which words will appear in bigrams and trigrams, the computer can do an excellent job of deciding which words match the phonemes that were uttered. The computer is programmed to make these decisions purely by knowing the computed probabilities, as opposed to understanding what the sentence is about. It does not need to have grammar rules or an

awareness of the situation. Thus, when we ask Siri a question, it can guess most of the words we are saying, as long as we are using words in conventional word orderings, which is most of the time.

However, knowing which words were uttered is just the first step. How does Siri answer our questions? It turns out that Siri is programmed to look for certain word patterns, much like the regular expressions created for a chatbot. Take the sentence, "What's the weather like?" or "Do I need an umbrella today?" Siri detects an expression with various weather keywords, and it sends a message to a special computer called a weather *server*. The message requests the weather for the current location. Remember, as a smartphone, Siri knows its current location. On the other hand, if Siri detects an expression about restaurants, it sends a message to a Yelp server. For movie reviews, it might send a message to a Rotten Tomatoes server. If Siri cannot match what you said to a pre-programmed pattern, it might send a message to Bing or Google to show whatever web page matches best with the words it detected. Sometimes people say questions just for fun, such as "Siri, what are you wearing?" or "What is the meaning of life?" These questions are called "Easter Eggs." Siri programmers have spent many hours analyzing thousands of Easter Eggs, and, just like a chatbot, the programmers provide Siri with a random clever answer.

After discussing systems such as Siri, the unit gives a short overview of how Google Translate works. When I taught the class, I brought up the online program and showed how it was able to produce an excellent translation, for example, of English to Spanish. Interestingly, the problem of translation is similar to speech recognition, except the Spanish translation programs needs to guess which Spanish word should match which English word. For translation, the program uses two types of corpora: 1) Documents that are translated between Spanish and English; and 2) Spanish corpora – millions of Spanish documents. From the first corpora, the

translation program computes statistics about which Spanish words most likely are translations of English words. As one could imagine, every English word could have a great number of possible Spanish translations. But from the second corpora—the one that just has millions of native Spanish sentences—the program guesses which Spanish words are most likely to go with one another. So the program finds the translation which has the highest probability of matching the English word in the context of the most likely Spanish words that occur with one another.

The results of this type of translation method can be quite good. But in many cases, statistics are not enough. For example, in Spanish the subject is often omitted from the sentence because everyone is assumed to know who is doing the action. In the sentence, *Mi madre le dijo a mi padre lo que quería para su cumpleaños* (My mother told my father what she wanted for her birthday) the word "she" would be left out. Also, the word for "her" in Spanish *su* could also mean "his," depending on the context. In this case, it is obviously "her." But without the common sense understanding about birthdays and birthday presents, it may be impossible for a computer program to reliably choose the correct pronoun. In this case, Google Translate mistranslates the Spanish to "My mother told my father what he wanted for his birthday." [29]

Unit 5: Computers and Common Sense: Teaching computers about human experience

Introduction

As the previous unit shows, despite the significant advances in NLP achieved with statistical techniques, there still has been little progress in programming computers to understand language in the way that John McCarthy envisioned, that is, to be able to understand language well enough to answer what we would consider obvious questions about a story or newspaper article. It still

[29] translate.google.com, May 1, 2015

seems that computers need to be programmed with commonsense data. Unit 5 (5 class sessions) discusses various ideas put forward over the past 40 years for accomplishing this as yet intractable task. The history of ideas for collecting commonsense data presented here is by no means complete.[30]

The unit begins with Schank and Abelson's idea of "scripts," which describe actions of everyday life in algorithmic terms suitable for programming (1977). Then it describes the Cyc project (Lenat, 1990), whose goal is to collect the commonsense knowledge needed to understand encyclopedia articles, and thus, understand documents and be able to learn from them. Then it presents Open Mind Common Sense (Singh et al., 2002), one of the earliest examples of crowd-sourcing. The project collects commonsense data from crowds—untrained people on the Internet— through various online games and activities. Finally, the Human Experience Project (Weltman, 2013) is a proposed method to collect detailed experiential data from amateur contributors, with some training, through stories about everyday life in the form of annotated comic panels.

By the end of this unit, students understand the advantages and disadvantages of each of these projects to collect commonsense data. Unfortunately, while interesting and instructive for AI and linguistics students, none of these projects has proven to be anywhere close to a solution to collecting the commonsense needed to understand even the simplest children's stories. However, studying these projects helps answer the central question of this course: why can't my computer understand me? Appendix 7 contains exercises for this unit.

[30] See Dreyfus (1992) for a more exhaustive (and pessimistic) analysis of commonsense modeling up to the early1990s.

Scripts (1977)

Suppose there is a story in which Sue goes to a restaurant to get a hamburger and comes home satisfied. As readers of this story, we understand a host of details that were not mentioned. We assume Sue ordered the hamburger, she paid for it, she ate it (or most of it), she might have used a napkin, there was someone who cooked the hamburger, there was someone who served it, and there was someone to collect the payment. A computer cannot possibly know these details, so Schank and Abelson (1977) proposed to create scripts of the most common life activities and gradually work up from there until a computer would have enough information to fill in the missing details of ordinary narratives. For a script such as "eating at a restaurant," the roles of the actors are identified (e.g., the chef, the waiter, the customer, the cashier) and typical actions in canonical order, such as the following: 1) The customer enters the restaurant; 2) The host greets the customer; 3) The host seats the customer at a table; 4) The host gives a menu to the customer; 5) The server arrives at the table; etc.[31] During class, we create scripts for a sit-down restaurant, and then we create a script for a fast food restaurant. Next we compare the two to understand one of difficulties of scripts: it is not clear when to create separate scripts (i.e., one script for a sit-down restaurant and one for a fast food) and when to create branches within scripts (i.e. one universal restaurant script with separate subcases for sit-down and fast food.) For example, in a fast food restaurant, the customer does not get greeted, the customer orders from a common menu, and the customer pays before the food arrives. Thus, if the fast food actions are part of a universal restaurant script, it would have to have if-statements in many places, as shown below:

[31] This is a simplified version of scripts. Schank's actual model include classifying actions according to various primitive categories, but this additional complexity was not necessary to understand the difficulties of creating scripts.

```
If fast food
Then
    The customer orders from the common menu
Else
    The host greets the customer
    The host seats the customer and gives the customer the menu
    The customer orders from the menu
```

As we find in class when actually trying to write a detailed script, dealing with different possible restaurant options in an algorithmic way soon becomes unmanageable. As with writing a complex computer program, we tried simplifying our task by creating small, cohesive chunks of activities that can be used by different scripts. For example, we tried a "Pay the bill" script. But even this seemingly small task quickly becomes complex when one considers all the ways that a bill can be paid, interactions with making change, and dealing with incorrect amounts. Furthermore, it is difficult to write these small scripts so that they are general enough to interact with a variety of other tasks. For example, a "Pay the bill" script differs widely in a fast food restaurant, where one pays at a cash register, and a sit-down restaurant where one discretely pays the server. In sum, writing "simple" life scripts becomes prohibitively complex when one has to manage all possibilities of what could happen. Although Schank's vision of a vast collection of detailed algorithmic life scripts never reached fruition, B. Li (2012) has developed a method that uses untrained workers on the Internet. Given an activity like going to a movie or going on a date, the workers create a simple list of actions that often make up these activities. Even though the lists are incomplete and lack detail, they may prove useful for improving NLP.

Cyc (1984)

Doug Lenat, first a Ph.D. student at Stanford in the 1970s and then a Stanford professor, wanted to figure out a better way to load programs with commonsense data. As Lenat says, the computer should be able to infer such thing as:

- When people die, they stop buying things
- Kerosene flows downhill
- When a bowl is overturned, its contents fall out

Rather than building a knowledge base about life activities or creating an artificial world with a limited number of actions, Lenat pledged to do the decades-long hard work, once and for all, of codifying the millions of bits and pieces of knowledge that comprise everyday common sense. With a small team of philosophers, linguists, and computer scientists, Lenat spent more than 20 years building an impressive "ontology," a catalogue of world knowledge organized into a hierarchy from general to specific and glued together by formal logic formulas. The system is called "Cyc," taken from the word "encyclopedia." It contains about ten million[32] hand-constructed facts and rules about history, everyday physics and chemistry, famous people, arithmetic, weather, and much, much more.

The part of Cyc referred to as the "upper ontology" contains the most general information. It contains the definition of a concept called a "thing." The most general concept in Cyc's hierarchy, everything in Cyc's ontology is a type of "thing." An "event" is a type of "individual thing," and Cyc has logical rules about events such as: If event A causes event B, then A must precede B. This rule is useful for reasoning about, say, a car accident and an injury. If there was an article about an accident causing an injury, the computer program would be able to infer that the accident occurred before the injury. Of course, this is so obvious to humans that we don't even think about it, but it is not to computers. At the other end of the spectrum in the lower ontology, Cyc has a great number of very specific rules about many topics. One of the specific rules about mammals, found in the lower ontology, is "If a mammal is exposed to the anthrax

[32] This figure is as of 2014 (Lenat and Durlach, 2014).

bacteria, the mammal becomes infected." Obviously, there are even more specific rules about people, dogs, movie directors, and thousands of other entities.

After more than three decades there is no indication that Cyc has come anywhere close to its original goal of being able to read and understand simple encyclopedia articles. The Cyc project still exists, but as of 2007, its focus seems to involve working with the language of specific domains like health care and anti-terrorism rather than general text understanding.[33] One of the bottlenecks blocking Cyc's reaching its original goal is that there seems to be no end to the how much commonsense data would be required for the type of general understanding Cyc envisioned. It seems that Cyc's strategy of engaging a small team of experts to catalogue human common sense is simply too slow and expensive.

Open Mind Common Sense (2002)

By the early 2000s, the Internet was a sensation, and AI researchers were wondering if the huge quantities of web pages it contained, as well as its online community, could be "harvested" for commonsense data acquisition. Push Singh and David Stork at MIT created a web site to collect informal folk knowledge from ordinary people, and the project was called Open Mind Common Sense (OMCS). Years before crowdsourcing became a common term to mean "getting work done by using free (or cheap) labor from people on the Internet," OMCS's target user was the unpaid volunteer who wanted to help AI. The most successful OMCS activity centered on useful binary relationships such as IS-A (e.g. a dog IS-A mammal), USED-FOR (e.g. a pot is USED-FOR cooking), and HAS-A (e.g. a bicycle HAS-A chain). Its web site offered a variety of easy-to-do activities to help volunteers input data using these simple lexical relationships. To understand the OMCS method of collecting data, the class plays a game where two-member

[33] Personal visit to Cyc, May 2007.

teams compete to guess words. One team member gives a clue, and the other makes a guess. But the clues have to use the standard OMCS binary relationships. For example, clues for the word "milk" might be, "It IS-A liquid. It HAS protein. It is USED-FOR drinking." In the successful online version of this game, Verbocity (von Ahn et al., 2006), the players communicate with each other on the Internet through fill-in-the-blank screens. As the players have fun, a program is in the background collecting the lexical data for use by OMCS. Of course, since ordinary people are inputting the data, errors and misinformation are to be expected. Nevertheless, the data were shown to be mostly correct, and the methodology is much cheaper and faster than using experts.[34]

Another OMCS proposed task was an online comic strip program. Volunteers create simple stories using comic frames with cute characters, captions, and dialog. People enjoy creating these stories. But since the captions and dialog are in free form text, the stories are difficult to analyze by an automated program, and so the commonsense data they might have cannot be easily collected.

OMCS showed the power of crowdsourcing to collect large amounts of commonsense data. However, it turns out that it worked best when the games involved simple fill-in-the-blank relationships. OMCS was unable to capture more complex commonsense relationships. There was no effective way for volunteers to tell a story, explain what was happening in the story, and explain character motivations. Yet this is the type of data needed for commonsense modeling.

[34] At about the same that OMCS was starting, another crowdsourcing project called MindPixel began. This project offered its contributors shares in future profits should the collected data lead to a commercial breakthrough in AI. MindPixel contributions consisted simply of yes/no questions on any topic. 1.5 million, often bizarre, questions were collected before the project's founder Chris McKintry committed suicide in 2006. As far as I know, the data has never been used for NLP. Tragically, Push Singh also committed suicide shortly after McKintry. (Kushner, 2008).

The Human Experience Project (2013)

OMCS's Push Singh and his colleagues recognized the value of collecting simple narratives for commonsense modeling (2004). Unlike scripts, which are a canonical set of events that describe a general *type* of activity, a story is about a particular event at a particular time. People are used to telling stories, and practically anyone can do it. Nevertheless, the various attempts to collect commonsense data from stories have not born fruit. The problem is that people naturally tell the interesting part of a story and leave out the commonsense explanations. For example, storytellers don't bother with mundane details about why a kid might want to eat a cookie. Everyone knows that cookies are delicious and sweet, and people, especially kids, like to eat such food, so it is not our habit to explain such matters. Or suppose we are writing a story about going outside to play. Who would bother explaining actions such as putting on a coat, walking to the door, grasping the door knob, and opening the door? However, these types of boring details are exactly what is needed if we want computers to have a human-level understanding of a simple story. Unfortunately, getting people, even experts, to articulate these details is extremely difficult.

The Human Experience Project (HXP) proposes a methodology that helps non-experts, with some training, provide these details. As an example, in class we discussed a simple story, based on material in (Weltman, 2013).

1) *Max was on the sofa, bored, all by himself. There was a pretty vase on a little side table. He went there and picked it up. He dropped it. Crash! This was fun!*

The class is asked to answer some commonsense questions:

- Where is Max? (He is probably in a living room or common area of a house.)

- At the start of the story, what is Max's body position? (He is probably sitting.[35])
- Where does Max go? (He goes to the table with the vase.)
- What does Max do to go there? (He first gets up from the sofa; then he walks to the table.)
- Why does he pick up the vase? (He is probably curious about it.)

None of these questions is explicitly answered in the text, but the typical reader answers them effortlessly. Surely, a prerequisite for understanding language is to be able to understand such a story as (1), and yet no one seems to have an idea of how to load this information into the computer.

The HXP proposal is for volunteers to create simple experiential narratives that would provide this type of data. The proposed narratives are comic strips like Figure 2. Each comic frame depicts a small action and has a caption that describes it. Critically, each comic frame is annotated with details about what is happening in the frame. For example, the location, the body position of the character, and the character's mental state. To help in this annotation process, HXP provides various software screens for input. Figure 3 shows the screen for the first frame "Max has nothing to do."

Through a series of fill-in-the-blank screens, the HXP program guides the user into adding the detail necessary to describe the frame. Part of this process involves answering questions similar to the Socratic Method. When a user adds a statement to the story, the HXP program asks the user to explain the statement.

[35] Of course, Max might be lying on the couch. Thus, a subjective confidence level of *possibly*, *probably*, or *definitely* is assigned to all statements.

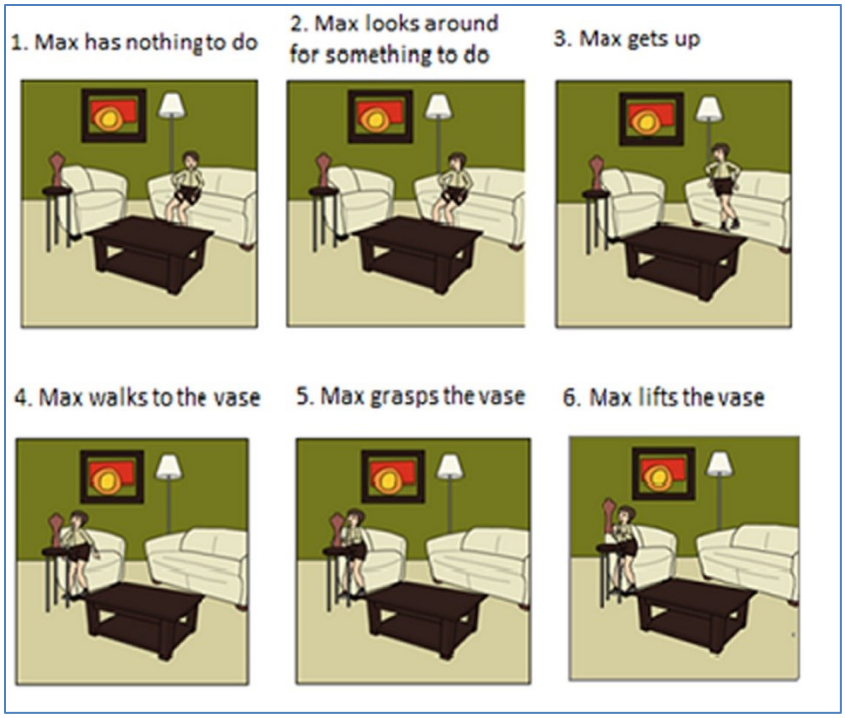

Figure 2: First six frames of "Max breaks the vase"

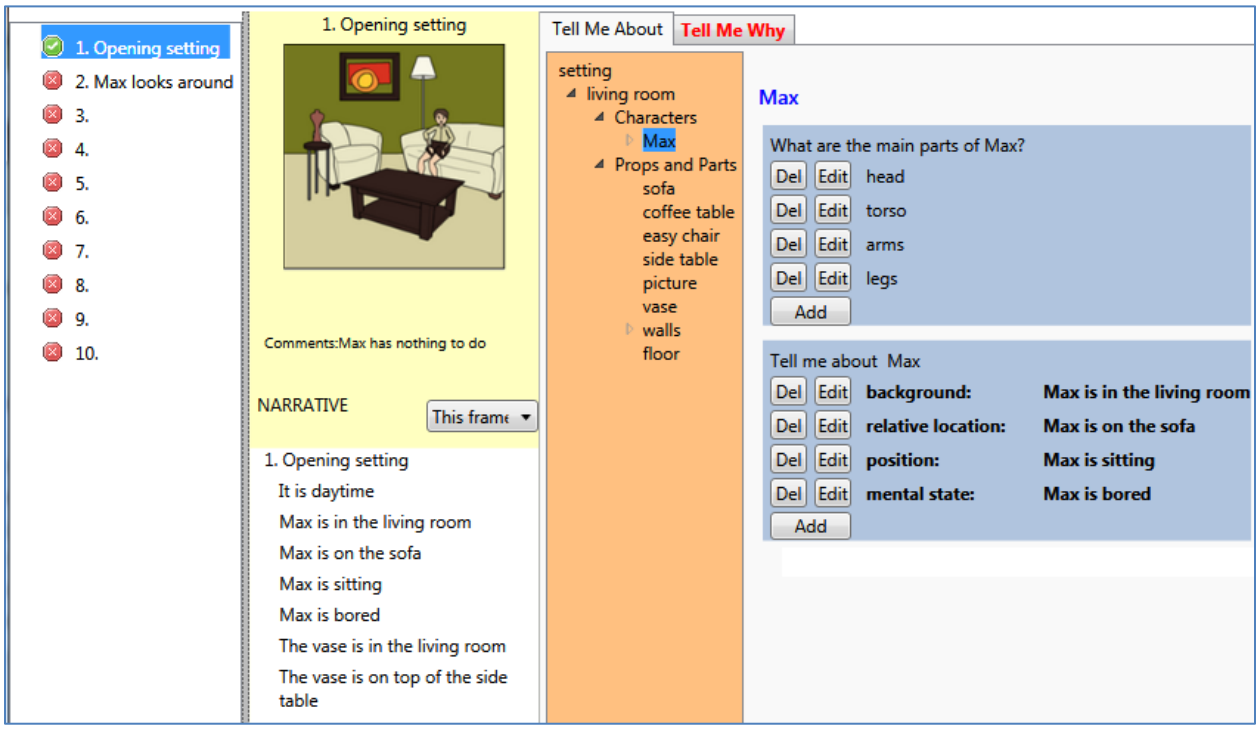

Figure 3: The opening setting after the user adds the background, character, and vase

Let's follow the process when the user is asked to explain why Max gets up in frame 3 of Figure 2 above. The user has to articulate Max's motivation in terms of what he has previously done. The user might try answering, "Because Max is bored." At this point, HXP's interactive Socratic Method would ask the user to confirm the following general rule of common sense:

2) IF a person is bored
   PROBABLY, the person gets up

Most people find that this is not a very good general rule. It is not really the case that people get up when they are bored. We recognize that more explanation is needed here. Most people recognize that Max stands up because he is curious about the vase. Each time they attempt to explain Max's action, HXP turns their answer around into a general rule, as in (3).

3) IF a person is curious about a vase
   PROBABLY, the person gets up

Once again, when presented as a rule in (3), the rule does not seem quite right. In another attempt, a person might say that Max gets up because he wants to examine the vase, which HXP would displays as rule (4).

4) IF a person desires to examine a vase
   PROBABLY, the person gets up

None of these seem to be correct, but through some training, the users realize that Max gets up for several reasons. He wants to examine the vase. He is sitting, and he is not near the vase. So that's why he gets up. HXP creates a rule like (5) to confirm.

5) IF a person is sitting
   AND the person desires to examine a vase
   AND the person is not near the vase
   PROBABLY the person stands up

Rule (5) seems to be a pretty good commonsense rule that explains Max's getting up in this action. During class, students practice doing annotations and creating rules.

After playing around with HXP, the students see that the HXP methodology helps amateurs, with some training, produce two types of data: 1) Detailed narratives and 2) General Commonsense Rules that explain the narratives. This data could be a valuable new resource for AI. However, HXP is just a proposal and proof of concept. It needs a lot of work to make it useable by the general public. Unfortunately, the majority of users (this class included) find the HXP process tedious. However, the good news is that there are a few people who find this work interesting. Thus, it may be possible someday to harness a small percent of the general population, which would be thousands of people, to do this type of work.

Unit 6: AI in Popular Culture

Introduction

Unit 6 (3 class sessions) gives students an opportunity to apply what they have learned about language and AI to popular culture. The unit examines the language and behavior of two well-known icons of AI: The HAL computer from *2001: A Space Odyssey* (1968), and the Data android from *Star Trek, The Next Generation* (1987-1994). In both cases, the AI entity is shown to be master of the most difficult aspects of language, and yet there are far simpler language skills, social skills, or technology that the AI mysteriously cannot handle. In the case of HAL, the main inconsistency is the fact that he behaves like an emotionless machine even when he is threatened. Yet programming the illusion of emotions and voice modulations would be trivial compared to HAL's other skills. And his voice is apparently based on a mechanical analog device like a record player rather than a digital speech synthesizer.[36] In comparison to HAL, the supposedly logical Data is far more illogical. Data easily handles difficult word sense

[36] Speech synthesizers, though not very good, had been around since the 1950s (Klatt, 1987), so it would not have been a stretch to assume that a machine in the year 2001 would use this technology.

ambiguities, conversational implicatures, and subtle humorous word play; yet, he has difficulty with bluffing at poker and simple puns.

Besides AI and language, this unit briefly touches on some other interesting aspects of these works, as well as science fiction in general. For example, the 1960s depiction of the year 2001 still has sexy stewardesses in 1960s hairdos to serve dinner in space flight and phone booths to make telephone calls, except the calls now have video. Thus, as has been pointed out by scholars such as (Freedman,2000), science fiction often reflects the society in which the movie was made as much as the futuristic society it portrays. Also, Turing's idea of intelligent machines remains a popular topic in AI; Commander Data's status as a conscious being and his rights is the theme in several Star Trek episodes.[37]

HAL from 2001, A Space Odyssey

HAL is a highly intelligent AI program that controls a space ship headed towards Jupiter on a secret mission, secret to even the humans that are flying the mission. Perhaps because of the contradictory situation in which HAL must hide information from his human commander, HAL's behavior starts to unravel, and he ends up going berserk.[38] Students were asked to watch the three-hour film on their own, and during class we showed several excerpts of the film and discussed them.

Because HAL is so famous for AI, many people have written about him, including three AI researchers that we covered in the previous unit: Roger Schank (innovator of scripts), Doug Lenat (leader of the Cyc project), and David Stork (one of the originators of OMCS). In Stork's

[37] The Turing test is still a hot topic in science fiction. The latest example is the sci-fi movie *Ex Machina* (2015) about a man who is must judge whether a newly invented android is intelligent.
[38] The movie does not give an explicit reason for why HAL starts to unravel. This is just one common interpretation.

edited book (1997), these researchers discuss HAL's use of language in the movie (Schank, 1997; Lenat, 1997), and I drew on their analysis, as well as my own, during the class discussion.

In one excerpt of the movie, a Mr. Amer from the BBC interviews HAL about his upcoming space flight (emphasis added).

> **Amer.** Hal, you have an enormous responsibility on this mission, in many ways perhaps the greatest responsibility of any single mission element. You are the brain and central nervous system of the ship, and your responsibilities include *watching over* the men *in hibernation*. Does *this* ever cause you any - lack of confidence?
>
> **HAL.** Let me put it this way, Mr Amer. The 9000 series is the most reliable computer ever made. No 9000 computer has *ever made a mistake* or distorted information. We are all, by any practical definition of the words, *foolproof and incapable of error*.

Schank notes that *watch* can have many different meanings: "You can watch TV, watch your baby, watch the clock, watch your back, watch it, watch how someone does something." Thus, HAL must be able to understand that "watching over the men" means to be responsible for their safety. Interestingly WordNet (Fellbaum, 1998), the most populate electronic resource for finding lexical associations and definitions, has an entry for *watch over* but it does not include the meaning used in this situation. Thus, the 2015 WordNet still isn't quite up to the task of providing suitable definitions for HAL.[39] Schank's point here is that HAL would need common sense to choose the most appropriate meaning of *watch* here. But since 1997 when Schank wrote this article, the problem of word sense disambiguity has become less difficult due to the statistical methods discussed in Unit 4. That is, "watch over" occurs often with "the men" in documents on the Internet, so it is conceivable that the ambiguity can be solved statistically without the need of common sense or context awareness.

[39] Not mentioned in Schank's enumeration of meanings, there is a more sinister interpretation of *watch* that has to do with spying and surveillance. This interpretation is particularly chilling because HAL later spies on the humans, reading their lips, as they attempt to have a private discussion.

Although not noted by Schank, Mr. Amer's question also has an interesting example of structural ambiguity in the use of the prepositional phrase "in hibernation." HAL must be able to understand that it is the men in hibernation and not HAL who is in hibernation. Resolving this ambiguity still requires common sense as one has to know that a person in hibernation is not capable of watching over anything.

Even more difficult, HAL must be able to resolve the reference to *this*. Does *this* refer to "watching over the men?" If so, HAL is being asked if he is nervous about watching over the men. But I think he is really being asked about the *responsibility* of watching over the men. There is no reason why someone would be nervous about watching something. The reference to *this* requires commonsense knowledge that having responsibility for a task is a burden that is prone to error. Yet HAL understands the question perfectly, as evidenced by his bragging about being incapable of error. Schank and Lenat point out that every intelligent being knows that errors are bound to occur. Thus, the fact that HAL is shown to have the obvious human frailty of false pride here is intriguing from a narrative point of view, even though it not consistent with HAL's intelligence.

The interview continues with a question revealing more of HAL's character (emphasis added).

> **Amer.** HAL, despite your enormous intellect, are you ever *frustrated* by your dependence on *people* to carry out actions?
>
> **HAL.** Not in the slightest bit. I enjoy working with people. I have a stimulating relationship with Dr. Poole and Dr. Bowman. My mission responsibilities range over the entire operation of the ship, so I am constantly occupied. I am putting myself to the fullest possible use, which is all, I think, that any *conscious* entity can ever hope to do.

The word 'people' could mean "people as a kind," as in "People need to feel loved." Or it could refer to specific people, as in "I think I hear people on the porch." HAL correctly interprets the

meaning to refer to the relevant people on his mission. The question also refers to the very complex emotion of *frustration*, a negative emotion, similar to anger, when one is prevented from reaching one's goal. HAL's response shows that he recognizes the implicit reference to the goal of taking care of the ship and depending on the people in his crew, Drs. Poole and Bowman, to achieve his goals. He also skillfully asserts indirectly that he is a conscious entity. That is, he uses a sophisticated presupposition where he makes reference to a conscious entity, and the reference only makes sense if it refers to himself. Indeed, under Turing's thesis, we should accept that HAL is a conscious entity because he appears so from his conversational abilities.

Later on in the movie, Lenat points out a fascinating example of commonsense understanding from a seemingly unimportant conversation between HAL and Dave, the ship's commander (emphasis added).

> **HAL.** Have you been doing some more work?
> **Dave**. Oh, few sketches.
> **HAL**. May I see them?
> **Dave.** Sure.
> **HAL.** That's a very nice rendering, Dave. I think you've improved a great deal. *Can you hold it a bit closer*?
> **Dave.** Sure.
> **HAL.** *That's Dr. Hunter,* isn't it?

As Lenat explains, the sentence "Can you hold it a bit closer is rife with ambiguity." HAL is using a conventional implicature to request that Dave move the sketch closer to HAL's camera-eye. That is, HAL never directly says "Please hold it closer. He asks a question, "Can you hold it closer?" A literal interpretation of this question might be "Yes, I can hold it closer." But, by convention, we know that this question really is a request. But using conventional implicatures is not that impressive for AI; it requires no common sense. On the other hand, HAL's understanding of how to use the deictic term 'closer' is impressive indeed. Lenat points out that moving something closer could possibly mean "closer to Dave," as in the example of a tennis

instructor asking a student to move the racquet closer, which would be closer to the student's body. HAL clearly possesses theory of mind; he can imagine how Dave will interpret the request, knowing it would not make sense to bring the sketch closer to Dave since HAL desires to see the sketch in more detail, so closer must refer to "closer to HAL's camera-eye." Next, HAL says, "That's Dr. Hunter." As Lenat points out, HAL is asking if 'that' is a *sketch* of Dr. Hunter, not literally if that *is* Dr. Hunter. Yet HAL's theory of mind capability allows him to understand that Dave will correctly interpret his reference.

In some ways HAL is a stereotypical AI robot in science fiction in that he apparently cannot handle logical contradictions. He has orders to keep the true mission to Jupiter from Dave, but these conflict with his orders to serve Dave. The variation here is that instead of the computer having the usual "Does not compute!" reaction with the requisite smoke, fire, and explosion, HAL slowly goes nuts.[40] But language and social behavior is full of contradictions, and someone as proficient as HAL with language should have no problem with a few logical anomalies such as keeping a secret from one's immediate supervisor in order to server one's higher commander. Also stereotypical, HAL's voice is very even and unemotional, and he speaks with a logical artificial formality and politeness. However, on a twist to this AI motif, and as a disparaging comment on humanity, the film portrays the humans as even more dull than the machine. "Kubrick and Clarke seem to want us to have stronger feelings toward HAL than we do toward the crew. The dull, robot-like astronauts sleepwalk through boring meetings and chat about ham sandwiches." (Stork, 1997).

[40] The phrase "Does not compute" was popularized in the television series *Lost in Space* (1965-1968), but it did not cause the robot's destruction. In contrast, several episodes of *Star Trek* (Original Series, 1966-1969) feature the cognitive dissonance computers face with contradictory or irrational human behavior, leading to circuit overload or destruction. Interestingly, in the movie *Desk Set* (1957), one of the more realistic early treatments of computers, Richard Sumner (Spencer Tracey) states that the computer cannot evaluate information; it can only repeat what has been fed into it. Later, to test Sumner's love, Bunny Watson (Katherine Hepburn) puts the computer on a path of destruction—not by ordering an impossible evaluation, as modern audiences might expect after this setup—but by the more prosaic (and realistic) method of removing a wire.

The director succeeds in making HAL more sympathetic than the humans, and his death scene is truly poignant; however, it is not consistent with the other AI technology in two significant respects: HAL's fear of death and his voice slow-down as he is being powered down. HAL says he is afraid to die, but shutting down a machine is usually not the end of the machine's life any more than shutting down my computer at the end of the day spells death to my PC. Normally, there is no fleeting soul of a computer. As with a word-processing document, all digital information can be saved and retrieved because the programmers know the exact format of the information. Every thought, every memory, every *bit* of knowledge that a HAL has can be restored. Of course, HAL might actually fear that once he is turned off, he will not be turned back on. But if that were the case, he would be pleading with Dave to make sure he will be eventually turned back on.[41] The departure from realistic AI technology could support an interpretation in which HAL has transcended beyond his programming and become a sentient being, perhaps through contact with the mysterious monolith.[42]

As HAL slowly dies, his voice gets slower and lower like phonograph record that is slowly winding down. In a film that in many ways is a model of science fiction Realism (quotidian detail, careful depiction of gravitational forces, prosaic conversations) to the point of being tedious, this depiction of HAL's voice is an unusual departure. HAL's utterances cannot possibly be pre-recorded to a phonograph-like analog device. He converses in real time with novel sentences that match the situation. Thus, his speech requires a dynamic speech synthesizer that would not get lower in frequency as it powers down. Of course it is no surprise that a director would take liberties for dramatic effect, and it is interesting to speculate on the artistic effect of

[41] In *Star Trek: The Next Generation*, "Thine Own Self" Lt. Commander Data, an android, gets radiation poisoning, loses his memory, and "dies." Fortunately, his body is found and he can be repaired without any apparent loss of personality. So, unlike *2001, A Space Odyssey*, the Star Trek writers decided, at least in this episode, that it is no tragedy to turn off and on a machine.
[42] Thanks to David H. Kirshner for pointing this out.

this directorial choice. This unit helps students better recognize and appreciate moments when poetic license is chosen over realism as an opportunity for critical interpretation.

Lt. Commander Data, from Star Trek: The Next Generation

Lt. Commander Data's interactions with his colleagues and friends make it clear that Data's programmers have solved the biggest problems of AI. Data has complex and subtle language skills. "He carries on conversations flawlessly. He enters into relationships with others. He has friends. He ponders, hopes, and is sometimes confused and bewildered. He even dreams." (Sennett, 1996). Yet, Data can't understand simple puns; he speaks with artificial formality, as exhibited by his lack of contractions; and he can't make a sneezing sound. There are several *Star Trek: The Next Generation* episodes centered on Data. In "The Measure of Man," (1988) Data plays poker and cannot understand what bluffing is. Considering the AI involved in Data's language skills, this lack of understanding is not consistent. Bluffing is just another example of playing the language game, imagining what the other person is thinking, and doing what is appropriate to get them to believe what you want them to believe. Even in the 1988 when the episode was written, it was understood that sophisticated language skills require the commonsensical understanding of the assumptions and goals of one's interlocutors.[43] It would be more realistic if Data had said he was programmed to be unable to tell a lie, but he doesn't say that. Instead he says he does not understand why someone would pretend to have a good hand when they don't. Data's naiveté here is part of a general pattern in which science fiction robots with full language capability still have to be taught human characteristics that are seemingly not needed for language understanding. With Data, the pattern is illogical indeed since Data is a

[43] That the ability to bluff, and to understand when others are bluffing, is a far easier problem then general language understanding is evident from the current state-of-the-art in the year 2015. While general language understanding is still a distant dream, poker-playing computers have succeeded in beating champion human players (Hamill, 2015).

soldier and would need a sophisticated theory of mind capability to interact with others and, especially, to deal effectively with an enemy. In "The Offspring" (1989), Data's android daughter, also with impressive language capabilities, cannot give a conventional answer to "How do you do?" This type of error is an example of a recurring theme in science fiction where the AI is portrayed as being unfamiliar with English idioms. But programming a machine to understand idioms is far easier than most of the other programming challenges for intelligent conversation. Understanding most idioms does not require common sense; it simply requires a glossary of phrases and their meanings.

In class we focused on the episode "Datalore" (1987) because it has several interesting language interactions worth analysis. Students watched the episode for homework, and then we screened parts in class. The first excerpt shows Data being unable to handle a simple word sense ambiguity, after Wesley hears Data practicing making a sneezing sound.

> WESELY: Have you got a... a "cold?"
> DATA: A cold what?
> WESLEY: It's a disease my mother says people used to get.

Word sense ambiguity is indeed difficult, but Data has no trouble with highly complex cases, as we will see below. The case here is simply is about deciding whether 'cold' is most likely a dangling adjective without a noun (highly unlikely) or noun (very likely). Although we are to understand that in the distant future, *cold* will not be part of vocabulary as a disease, computers can easily store vocabulary words of millions of words, so a realistic AI program would still contain arcane words. Exhibiting a common science fiction pattern, Data's difficulty is more in line with the types of jokes that occur when a person is unfamiliar with English vocabulary and cannot understand a pun. Thus, the writers conflate, for the purpose of entertainment, what is difficult for foreign language speakers with what is difficult to program.

As the ship approaches the destination planet, Commander Riker asks Data, “Want to take her into orbit?” Data has no problem understanding that “Want to” means “Do you want to.” And he has no problem resolving the reference to ‘her’ to the space ship, which of course, means understanding that the feminine pronoun can refer to a ship, and that the ship is ready to be taken into orbit.

We learn that the ship is in orbit around the planet where Data was first discovered. Captain Picard engages in some playful banter with Data and First Officer Riker:

> PICARD: This is an exciting moment for you, Data. I'm tempted to lead the away team down myself, except for the fact the first officer would object…

There are some subtle nuances in this statement for Data to understand. First, what does “this moment” refer to? It could refer to any number of moments: this particular moment in time when the Captain is speaking, the span of time they started orbiting the planet, the span of time between breakfast and now, and any other relevant time. In this case, the relevant moment is the moment when Data will return to the place where he was discovered. It takes a lot of experiential knowledge to realize that returning to a place where one was discovered is comparable to returning to one’s home, that returning to one’s home is special, and thus it would be considered a special moment for Data. Data also knows that the captain is not really serious about leading the away team, but rather, that the Captain is saying to Riker that he, the Captain, is allowing Riker to lead the away team. When a person says, “I’m tempted to do X,” he usually means that he is not going to do X. Also normally the captain would refer to Riker by name since he is in Riker’s presence, but here he uses Riker’s formal title when he says,“the first officer would object.” Thus, he is flouting convention here and signaling by conversational implicature that he is not being serious. For Data to fully participate in this joke is to wield considerable skill with language.

When they get close to the destination planet, Data says, “I could say "home, sweet home," sir... if I understood how the word ‘sweet’ applies.” We are to understand that Data does not know the possible meanings of sweet. One interpretation is that the writers are making a statement that machines cannot truly understand how a home can be sweet because they cannot feel sweetness. But this contradicts the later events when Captain Picard states that we are all machines, and Data is “merely a different variety of machine.”

> PICARD (to Data): We know that you're as "alive" as any of the rest of us.
> (to the others)
> If you find it awkward to be reminded that Data is a "machine"... you might remember that the rest of you are merely a different variety of machine... in our case, electro-chemical in nature

This stance on Data’s aliveness clearly goes further than Turing’s idea of functional intelligence. According to Picard, humans and androids like Data are equally alive. Humans are an electro-chemical machine, whereas androids are just electric. Indeed, other Star Trek episodes such as In “The Measure of Man” (1988) explore the idea that androids have equal rights with humans.

As a final note, this episode reveals why Data speaks in an unnecessarily formal way, not using contractions and not making jokes, even though clearly his creators could have programmed him to have a more informal personality since they already solved much more difficult language problems. It turns out that Data’s predecessor android, named Lore, was programmed to be more like a human. Lore was able to make jokes, and he spoke in a much more relaxed fashion. But he was “too human” and became aggressive and conniving. Thus, an android’s language habits are seen to be indicative of a deeper and more dangerous level of humanness. For Turing, language was a measure of intelligence. For *Star Trek’s* writers, language is a measure of humanity.

Unit 7: Group presentations on invented languages or AI-related literature

Unit 7 (5 class sessions) is an exploration of invented languages. In one sense, the unit has very little to do with NLP. However, many people consider computer languages to be artificial languages, so a goal of this unit is to compare programming languages to other artificial languages. On this topic, the message is very clear. While programming languages have vocabulary and syntax, they are not in any other respect like natural language. Although programming languages can be expressive, they have specific and very limited vocabulary. Most importantly, they have absolutely no ambiguity of sense, structure, or context. Every word and punctuation has a well-defined context and there is only one meaning possible. They are really just long mathematical formulas, and the politicians who want to allow a programming language to serve as credit for a foreign language (Wilson, 2014) are not making a comparable substitution.

Another goal of this unit is to introduce students to linguistic topics such as phonetics, phonology, and cultural linguistics. Several students were excited to be studying about languages from books and movies, so learning about invented languages was a hook to get them interested in linguistics.

Right after midterms, before we had finished Unit 5 on Collecting Commonsense Data, I gave a short lecture on invented languages based mostly on material from *In the Land of Invented Languages* by Arika Okrent. After this tour through various invented languages, I instructed students to form a group of two or three people and to choose a language for an oral presentation. The instructions for this group project are in Appendix 8. Thus, while we worked on Units 5 and 6, students were to be preparing for their 15-minute group presentation. The last three weeks of class mostly comprised these presentations. The topics that students chose varied.

We had presentations on the phonetics of Klingon, the syntax of Atlantean (from the movie *Atlantis: The Lost Empire*), Doktraki from the television series *Game of Thrones*," La'adan (the invented language to remove bias against women), and the Elvish language Tenwar from *The Lord of the Rings*. Also, students were allowed to present on topics from other units in the class. One group did a presentation on the life of Alan Turing and another analyzed language and AI in the film *Her* (2013).

Conclusion

This chapter presents the material for the seven units of the course. The first unit introduces Alan Turing, the computer pioneer, war hero, and tragic victim of England's laws against homosexuality. The fascinating account of Turing and his theories about AI and language motivates the second unit, the introduction to programming and NLP. The third unit explores what it would mean for a computer to have a human-level understanding of language and presents concepts from semantics and pragmatics to analyze the simplest possible natural language: stories for very young children. In doing this analysis, students realize that human understanding of language, even in the most straightforward text, depends greatly on commonsense knowledge gained through life experience. Nevertheless, current NLP programs such as Google and Siri appear to understand language, so the fourth unit briefly describes the statistical method of NLP. In this method, a program does not use grammar, context, or common sense at all. Rather, it processes words according to their statistical frequencies and the probabilities of different word combinations derived from billions of sentences on the Internet and other large corpora. But the statistical method is limited, and it seems that commonsense data would be necessary to have a deep semantic understanding. The fifth unit looks at various methods for loading commonsense data into a computer, none of them successful so far. At this

point, students will have grasped why programming NLP is so difficult. The sixth unit applies this knowledge to the analysis of language in AI as it appears in popular culture, and we note interesting inconsistencies when sophisticated AI entities have difficulties with relatively easy language tasks. The final unit introduces invented languages, comparing them to programming languages, and exposing students to a variety of linguistics topics as they learn about the language or AI topic that most interests them.

A large amount of the material developed for teaching this course is collected in the appendices. Appendix 1 is the course syllabus. Appendix 2 is the class schedule, as it was taught in Fall 2014. Appendices 3-8 have material for units 1-6, listing the unit objectives, homework exercises, and other teaching material. Appendix 9 has the student instructions and rubric for group presentations on invented languages used for unit 7. Finally, Appendix 10 lists the surveys and data used to evaluate the course after teaching it in Fall 2014.

CHAPTER 4: COURSE EVALUATION AND RECOMMENDATIONS

Introduction

I taught this class in the fall of 2014 and asked the students to fill out a survey on the course material on the last day of class. Later, I did two follow-up surveys to get more details. The registration for the class was filled to the maximum of 20 students. All students stayed in the class for the duration, and 85% got an A or B. Survey results indicate that most of the students found the course material valuable. Disappointingly, more than 40% of the class rated the computer programming part of the course as having a value of below average. Despite my efforts to make programming details palatable to humanities majors, many found the technical minutiae tedious and complex, although more than 30% rated this material above average. Somewhat surprising, more than 35% of the class negatively rated the unit on AI and popular culture. Most of the negativity appears to stem from the requirement to watch *2001, A Space Odyssey*, which admittedly can try one's patience. On the other hand, the survey indicated quite favorable impressions of the other units, especially Computational Semantics and Invented Languages, and the data show that taking the class resulted in at least some students feeling more confident about the idea of interacting with computer programs and programmers. In future iterations of this course, I would either reduce the number of sessions on Python programming, or I would show more practical examples of programming to provide further motivation to learning the details.

The rest of this chapter discusses the details of the survey and results. It first describes the format of the surveys. Then it goes through each unit, summarizing student feedback and offering recommendations on how the unit could be improved.

Description of the Surveys

The three surveys are shown in Appendix 10. The main survey is Survey 1, which is divided into three parts. Part I asks the students to evaluate the value of the unit's material and the time spent on each unit. Part II asks students to rate the value of the homework projects. Part III assesses the level of difficulty of the course and the work load. Survey 2 was a follow-up to gather additional information about students' reactions to the unit on programming. Finally, Survey 3, conducted three months after the semester ended, assesses whether the goals of this class were achieved and also collects more information about why some students did not like the unit on popular culture. 19 of 20 students completed Surveys 1 and 2. In contrast, only 7 of the 20 students responded to Survey 3, probably because three months had elapsed and the survey was to be filled out online rather than in class. The numerical data from Survey 1 and Survey 3 is shown in its entirety in Appendix 10. The survey results that involve comments are not shown to protect anonymity, but the general ideas of the comments are discussed in the summaries below.

Survey Results

Figure 4 shows a summary of how the students assessed the value of the units. Students rated the value of each unit on a scale of 1 to 5, in which the values are 1 "No value," 2 "Some value", 3 "Average value", 4 "Very valuable," and 5 "Extremely valuable." For ease of presentation, the table combines values 1 and 2 to "Below Average" and combines 4 and 5 to "Above Average." In a similar manner, Figure 5 summarizes the students' answers to the question about time spent. Possible answers were 1 "Not nearly enough," 2 "Too little," 3 "Just right," 4 "Too much," and 5 "Way too much."

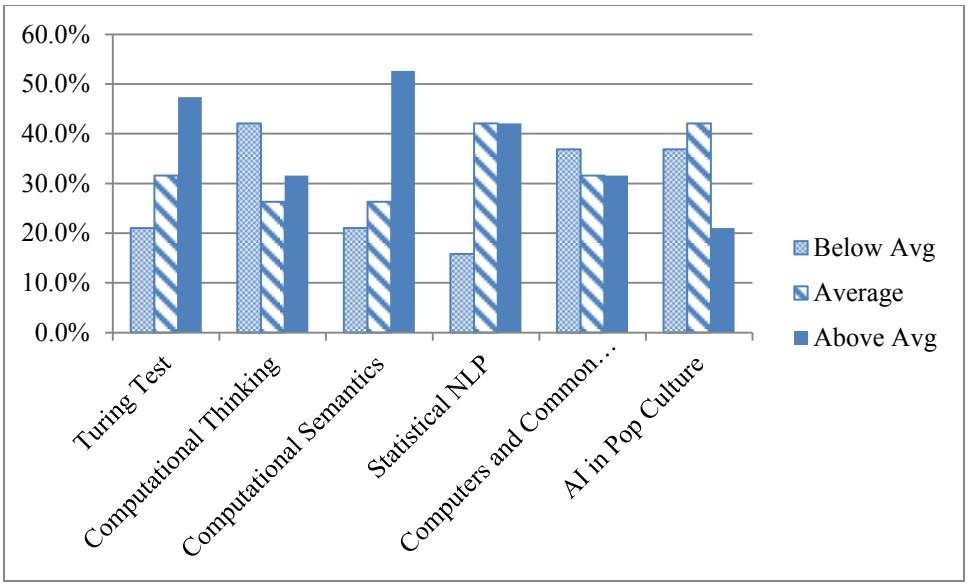

Figure 4: Student responses from Survey 1 showing how they rated the value of each unit. More than 50% of the students rated Computational Semantics as having a value above average. More than 40% of the students rated Computational Thinking below average.

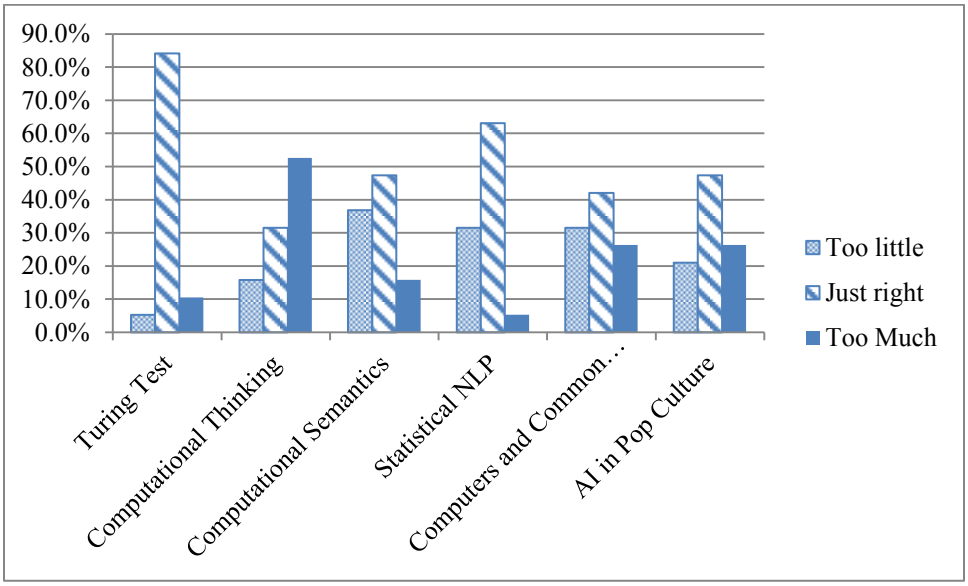

Figure 5: Student responses from Survey 1 showing how they rated the time spent in each unit. More than 80% of the students felt the time spent on the Turing Test unit was just right. In contrast for Computational Thinking, only about 30% felt the time was just right, and more than 50% felt that too much time was spent in that unit.

In Figure 4 above, ideally students would rate each unit as having above average value, and in Figure 5 the ideal time spent on the unit would be "just right." [44]

Unit 1: The Turing Test

For Unit 1: The Turing Test, Figure 4 shows that 21% of the students ranked it below average, 32% ranked it average, and 47% ranked it above average. Thus, the class overall considered Unit 1 to be above average, and I interpret this to mean this unit was very successful in engaging students. I had expected as much since the unit is about the interesting life of Turing and his fascinating theory of machine intelligence based on language. In addition, almost 84% of the students felt that the time we spent in this unit was just right, as shown in Figure 5.

Unit 2: Computational Thinking

Unit 2: Computational Thinking is the unit involving programming in Python. Figure 4 shows that it received an unfavorable rating, below average value, by 42% of the students, an average rating by 26% of the students, and an above average ranking by 32% of the students. Also, as shown in Figure 5, more than 50% of the class felt that too much time was spent on it. Thus, a large minority of the class had negative feelings toward this unit. In Survey 2, conducted after the final exam, I asked students for additional comments about this unit. Several commented that the material was too complex and boring. On the other hand, a few of the students said that they would have liked to have done more advanced work. However, given that more students were not interested in programming details, I would recommend a different approach the next time this course is taught. According to Dennis Tenen, one of the instructors in Columbia University's innovative new course in programming for the humanities (Wood and

[44] I interpret a rating of "Too much time" to mean that the students did not enjoy the material. Conversely, "Too little time" means either the students wanted to move more slowly, or they wanted more in-depth study.

Bix, 2014), students are more motivated to learn about programming if they can analyze their own texts.[45] Therefore, I would create some programming exercises in which students analyze vocabulary, style, or other language elements of their own works. For example, students could write a simple script that counts the most frequently used words in one of their own papers from another class. Or students could write a program that counts the use of passive voice (a "be" verb followed by a past participle.) Since the comments also indicate that there may be an audience for a more in-depth programming class for humanities majors, I would also recommend development of full semester class on programming, perhaps similar to the one at Columbia.

Unit 3: Computational Semantics

Students ranked Unit 3: Computational Semantics relatively highly. As Figure 4 indicates, while 21% ranked it below average in value and 26% ranked it average, 53% thought it above average. The material in this unit includes a survey of semantics and pragmatics topics, so linguistics issues clearly resonated with this student population. This positive reaction is not surprising for a linguistics class, but it indicates that future iterations of this course could go into more depth in these topics, particularly since Figure 5 shows 37% of the students would have liked to have spent more time on this material and only 16% thought we spent too much time here.

Unit 4: Statistical Natural Language Processing

Unit 4: Statistical Natural Language Processing was an experiment. It contains the most mathematically oriented material of the course, and I did not know if students would be interested in learning how probabilities and number crunching provide the illusion of language

[45] Personal conversation, Feb 18, 2015.

understanding. It turned out that students were indeed interested. As Figure 4 shows, more than 41% ranked it above average, and only 16% ranked it below average. Furthermore, Figure 5 indicates 32% would have liked to spend more time on this subject, versus only 5% who thought it was too much time. Based on this feedback, future iterations of this course could explore this material in more detail, possibly adding discussion on how to gather statistics from various well-known corpora such as the Penn Treebank.

Unit 5: Collecting Common Sense

For Unit 5: Collecting Common Sense, there was no clear consensus. 32% ranked it below average, 42% ranked it average, and 26% ranked it above average. Similar numbers apply to the amount of time spent. So it seems that some students liked the material and some did not, but I would not recommend any changes here based on this feedback.

Unit 6: AI in popular culture

The reaction to Unit 6: AI in popular culture surprising. I had thought almost all of the students would enjoy looking at classic sci-fi works in popular culture and analyzing the AI elements. However, 37% gave it below average, and only 21% ranked it above. To get more information, I asked students to provide more feedback in Survey 3, conducted online three months after the class ended. Several students commented that they hated *2001, A Space Odyssey* or that they liked the AI issues but not the movie. In the future, I would handle the movie differently. Rather than asking students to watch this three-hour, challenging movie on their own, I would spend more time in class showing excerpts from key scenes and discussing their relevant issues in AI and language.

Unit 7: Presentations on Invented Languages

Figure 6 below shows the survey results regarding the presentations on invented languages. Allowing students to give presentations on invented languages is an excellent opportunity for students to learn about wide variety of topics, but unless the presentations are lively and polished, they are not much fun to listen in. I gave specific instructions to the students to rehearse and polish their presentations, and most of the students did a good job. The survey asked one question about the value of both giving oral presentations and listening to other students' presentations. I was pleased that the students gave high marks for both. More than 50% rated doing presentations above average, and about the same rated listening to them above average.

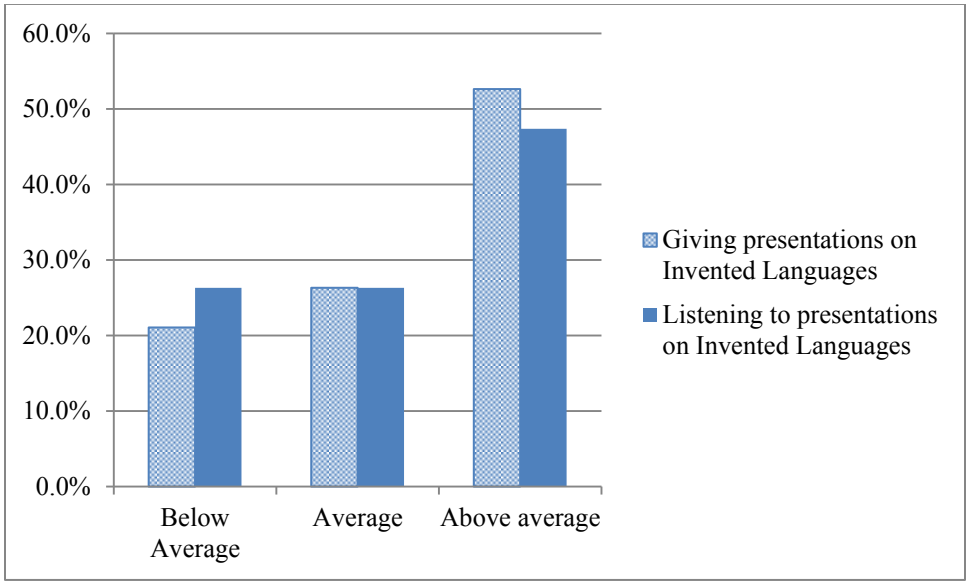

Figure 6: Value of giving presentations on invented languages. More than 50% rated doing presentations as above average, and a similar number rated listening to the presentations above average.

Effectiveness of Class

Survey 3 attempts to gauge the effectiveness of the class. It asks students to self-assess their understanding of a computer program both before they took the class and after having taken the

class. It next asks students to imagine that they have a job working with a computer programmer. What would have been their comfort level in this collaboration before they took the class, and what would be their comfort level after having taking the class? For the "before" and "after" ratings, students chose among Very Low, Low, Medium, High, and Very High. Since only 7 of the 20 students responded to the online survey, there is not enough data to gauge the effectiveness of the class with all the students. However, Figure 7 and Figure 8 shows a summary of data for the seven students that did respond. With regards to understanding programming, 5 of the 7 students reported a jump of at least one place after having taken the class. One of these students clearly did not enjoy the material, reporting only going from only Very Low to Low. (This same student later reported a *decreased* interest in programming after taking this course.) The other two students started relatively high before taking the class and did not change. Nevertheless, we can conclude from this data that at least some students felt that they advanced their learning of computer programming in this class.

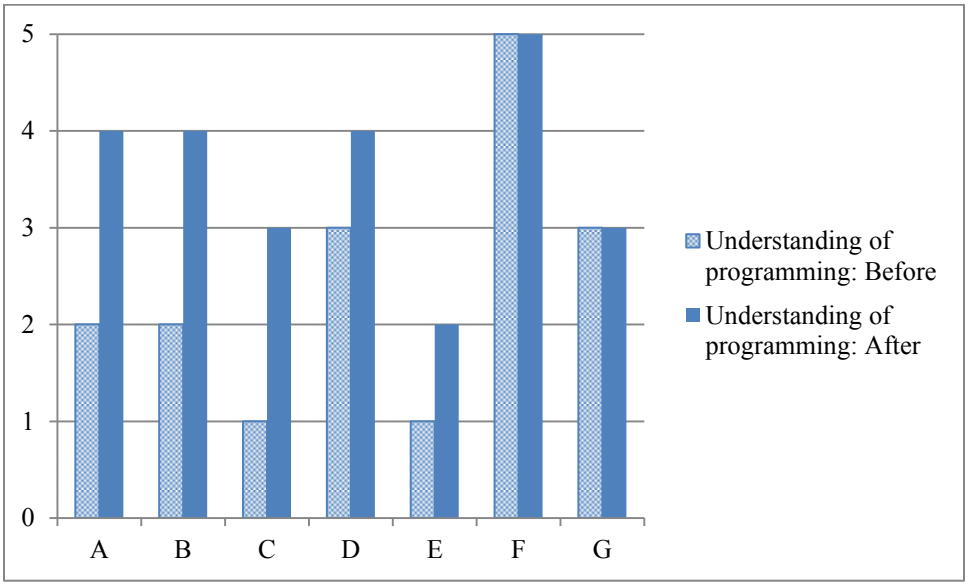

Figure 7: Student' self-assessment of their understanding of programming: Before taking the class and after taking the class. The first two students moved from Low (2) to High (4). In total, five students moved up at least one level.

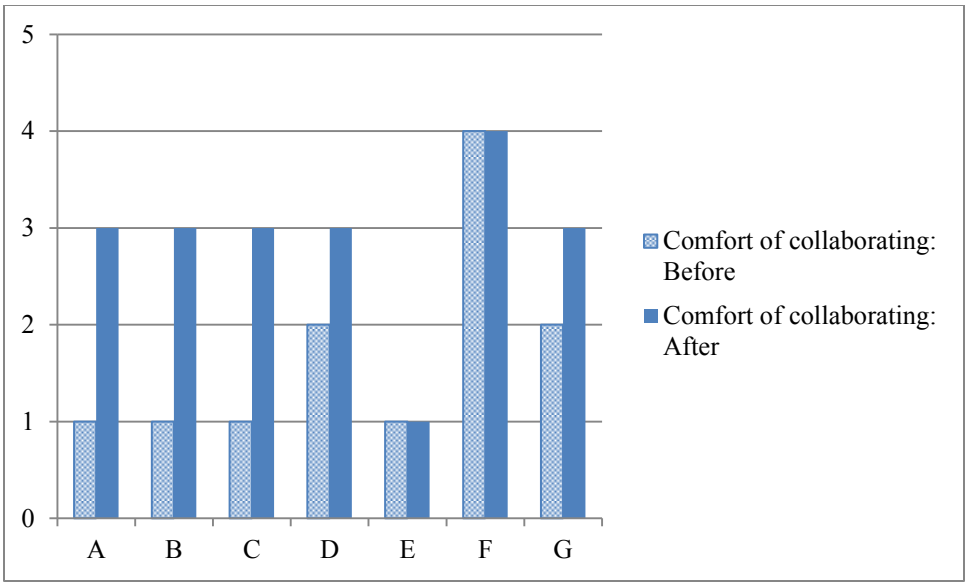

Figure 8: Students' self-assessment of their comfort level with the idea of collaborating with a computer programmer: Before and after taking the class. Five of the seven students moved up at least one level

Figure 8 shows the students' self-described comfort level with the idea of collaborating with a computer programmer. The first three students in the table moved from Very Low to Medium. Two other students moved up one level. Again, one student had a negative reaction to the material and did not change. The remaining student started Very High and did not change. Again, with less than a third of the class responding, the only conclusion that can be made from these results is that at least some students felt that they reached a better understanding of programming and a higher comfort level working with programmers by taking the class. In addition to students' self-assessed results, the vast majority of the students understood basic coding well enough to decipher an unfamiliar 20-line Python function containing a double while-loop, which was given to them on the mid-term exam. In this respect, they achieved the basic level of code understanding for which I had hoped.

Conclusion

After teaching the class one time, class surveys indicate most students were engaged by the course material; however, there are several areas of improvement for the next time the course is taught. Namely, a significant minority of the students would have preferred more of a conceptual approach to the unit on programming Python rather than learning the details of a programming language. One possible change would be to make the material slightly less technical, leaving out details of Python variable assignments and function calls. However, I believe that students would appreciate the details if they could see a practical benefit to learning them. The innovative approach with the new humanities computer programming course at Columbia University is to have students write scripts to analyze their own writings to identify overused vocabulary or trite phrases. Thus, students get not only a conceptual understanding of programming but take away a useful tool for improving their own work. The next iteration of this class could take a similar approach. Another area of improvement would be to spend more time in class showing excerpts from the *2001, A Space Odyssey* to help students appreciate the interesting AI issues. Finally, the course could spend more time with statistical natural language processing and corpus linguistics, since students would have liked more of this material.

CONCLUSION

Just as a basic understanding of an automobile makes us able to feel more comfortable around a car mechanic, the non-STEM professional needs to have a basic understanding of computer programming not only to collaborate but also to more effectively manage everyday life. Note, however, that the analogy of computer skills to car skills is a weak one. As John McNaughton points out, unlike computer programs, "...cars don't run the world, monitor our communications, power our mobile phones, manage our bank accounts, keep our diaries, mediate our social relationships, snoop our social activities and even— in some countries—count our votes." (McNaughton, 2012). Those without knowledge of computational processes will be increasingly dependent on the elites who do. As Naughton so colorfully puts it, members of the public who are ignorant of programming are like "hamsters for the glittering wheels of cages built by (Facebook's) Mark Zuckerberg and his kind."

As this thesis demonstrates, I have designed a class to teach basic code literacy to humanities majors using AI and natural language processing as vehicles to motivate learning. The course begins with Alan Turing's famous theory of machine intelligence, proposing that the key indicator of intelligence is the ability to hold a conversation. Of course, implicit in this proposal is that a computer has to be able to understand language. The course then teaches enough coding skills so that students can start to grasp the difficulty of writing algorithms and collecting data to simulate language understanding. Drawing on well-known linguistics phenomena in semantics and pragmatics, the course guides students through some of the most difficult problems for a computer, and students understand that common sense and life experience seem to bear a significant role when humans process language. Nevertheless, NLP programs like Siri and Google appear to understand speech and be capable of intelligently handling a variety of

language tasks like question answering and language translation. This course discusses in non-mathematical terms how such programs use statistical probability and word frequencies to simulate understanding. But these techniques alone are not capable of providing the human-level language understanding envisioned by AI enthusiasts and science fiction. It still seems that we must program computers to have common sense. The course then covers several attempts at overcoming the challenge of collecting commonsense data for computation, but the conclusion is that computer programs have not made much progress in reaching this goal. Using this newly found awareness of AI and the limitations of NLP, the class analyzes AI as it is presented in popular culture with an increased knowledge of when authors and directors take liberties with realistic portrayals of AI.

The class concludes with a unit on invented languages in which students explore the linguistic structure, phonology, and culture of languages such as Esperanto, Klingon, and Elvish. Programming languages are also considered by some to be invented languages, but this class teaches that programming languages are really just extended math formulas. The meaning of each word in a program is unambiguous; all references are precisely defined; all phrases have only one possible structure. In contrast, even languages with a small vocabulary or languages that attempt to attach specific meanings to all words have ambiguity that arises from how the words and syntax interact with the current context. In other words, languages used for personal communication, both invented and natural, depend on the current situation, the tone of voice of the speaker, the status of the speaker in the culture, the events that occurred before that are relevant to the utterance, the expectations of the listeners, and the assumptions of the audience, among other things.

Thus, virtually all of the interesting and complex issues studied in semantics and pragmatics occur with all natural languages, as well as with invented languages for personal communication, but not for programming languages. Indeed, programming languages must be tightly constrained so that a computer will understand them. One of the most important outcomes of becoming code literate is understanding the strengths and limitations of computer processing. Computers can tirelessly sift through vast amounts of text and data in search of patterns—as long as a programmer can provide suitable data and articulate what pattern to search for. The patterns and data that will allow computers to interpret natural language in context are still to be discovered.

REFERENCES

2001, A Space Odyssey. Dir. Stanley Kubrick. Perf. Keir Dullea, Lockwood, Gary and Sylvester, William. 1968.

Amlen, Deb. "Our Interview with Turing Test Winner 'Eugene Goostman'." *YAHOO!Tech* 13 June 2014.

Bach, Deborah. "How to interest girls in computer science and engineering? Shift the stereotypes." *UW Today* 11 February 2015. 14 February 2015. <http://www.washington.edu/news/2015/02/11/how-to-interest-girls-in-computer-science-and-engineering-shift-the-stereotypes/>.

Barker, Lecia J., Kathy Garvin-Doxas and Eric Robers. "What Can Computer Science Learn from a Fine Arts." *SIGCSE'05*. St. Louis, MO: ACM, 2005.

Beacock, Ian P. "New Stanford course brings Silicon Valley to the humanities classroom." *Stanford News* January 6 2015.

Bidwell, Allie. *US News & World Report* January 27, 2015.

Bird, Stephen, Evan Klein and Roper Roper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

Bradshaw, Tim. "Scientists and Investors Warn over AI." *The Financial Times* January 15, 2015.

Breaking the Code. By Andrew Hodges and Whitemore, Hugh. Dir. Herbert Wise. Perf. Derek Jacobi. 1997. TV Movie. <www.youtube.com/watch?v=S23yie-779k>.

Brennan, S.E., M.W. Friedman and C. J. Pollard. "A Centering Approach to Pronouns." *Proceedings of the 25th ACL*. 1987.

Brinton, D.M., M.A. Snow and M.B. Wesche. *Content-based second language instruction*. New York: Newbury House, 1989.

Bui, Quoctrung. "Who Studies What? Men, Women And College Majors." *Planet Money*. National Public Radio, 28 October 2014. 19 May 2015. <http://www.npr.org/sections/money/2014/10/28/359419934/who-studies-what-men-women-and-college-majors>.

Charniak, Eugene. "Inference and Knowledge, Part I." (eds.) Charniak, Eugene and Yorick Wilks. *Computational Semantics: An Introduction to Artificial Intelligence and Natural Language Comprehension*. Amsterdam, New York, Oxford: North-Holland Publishing Company, 1976.

—. *Toward a model of children's story comprehension*. Technical Report AITR-266. Cambridge, MA: Artificiial Intelligence Laboratory, Massachusetts Institute of Technology, 1972.

Courage, Katherine and Kathy Baxter. *Understanding Your Users: A Practical Guide to User Requirements Methods, Tools, and Techniques (Interactive Technologies)*. San Francisco: Morgan Kaufmann Publishers, 2005.

"Datalore." *Star Trek: The Next Generation (Television Series)*. Teleplay Robert Lewin and Gene Roddenberry. Story Robert Lewin and Bill Smith. Dir. Rob Bowman. Paramount Pictures Corporation, 1987.

Diehl, Mike. "Teaching Programming Skills to Children with Logo." *Linux Journal* April 16, 2009.

Dreyfus, Hubert L. *What Computers "Still" Can't Do: A Critique of Artificial Reason, Revised edition*. Cambridge: MIT Press, 1992.

Dubrow, Aaron. "Just in time: Tips for computer science teachers when they need it." *National Science Foundation Discoveries* (2015).

Dubrow, Aaron and Katie Hendrickson. "Artful coders." *National Science Foundation Discoveries* (2015).

Fellbaum, Christiane. *WordNet: An Electronic Lexical Database*. Cambridge, MA.: MIT Press, 1998.

Freedman, Carl. *Critical Theory and Science Fiction*. Middletown, CT: Wesleyan University Press, 2000.

Grabe, W. and F.L. Stoller. "The content-based classroom: Perspectives on integrating language and content." *Content-based instruction: Research foundations*. Ed. M.A. Snow and D. M. Brinton. NY: Longman, 1997. 5-21.

Hamill, Sean D. "Who's got the upper hand? Poker computer program pits man against machine." *Pittsburgh Post-Gazette* 24 April 2015.

Her. Dir. Spike Jonze. Perf. Joaquin Phoenix. 2013.

Holabird, Katherine. *Angelina and the Butterfly*. New York: Penguin Young Readers Group, 2006.

Johnson, Bernadette. "How Siri Works." *HowStuffWorks.com* 6 February 2013. <<http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/siri.htm> >.

Kelleher, Caitlin and Randy Pausch. "Using storytelling to motivate programming." *Communications of the ACM* 50.7 (2007): 58-64.

Klatt, D. "Review of Text-to-Speech Conversion for English." *Journal of the Acoustical Society of America* 82.3 (1987): 737-93.

Kushner, David. "Two AI Pioneers. Two Bizarre Suicides. What Really Happened?" *Wired Magazine* 18 January 2008.

LeFevre, Jo-Anne, Alison G. Kulak and Stephanie L. Heymans. "Factors influencing the selection of university majors varying in mathematical content." *Canadian Journal of Behavioural Science/Revue canadienne des sciences du comportement* 24.3 (1992): 276-289. <http://dx.doi.org/10.1037/h0078742>.

Lenat, Douglas B. "From 2001 to 2001: Common Sense and the Mind of HA." *HAL's Legacy: 2001's Computer as Dream and Reality*. Ed. David C. Stork. Cambridge, MA: MIT, 1997.

Lenat, Douglas B. and Paula J. Durlach. "Reinforcing Math Knowledge by Immersing Students." *International Journal of Artificial Intelligence in Education* 24 (2014): 216-250.

Lenat, Douglas B., et al. "Cyc: toward programs with common sense." *Communications of the ACM* 33.8 (1990): 30-49.

Levinson, Stephen C. *Pragmatics*. Cambridge: Cambridge University Press, 1983.

Levitz, Jennifer and Douglas Belkin. "Humanities Fall From Favor." *The Wall Street Journal* June 6, 2013.

Marcus, Mitchell P., Beatrice Santorini and Mary Ann Marcinkiewicz. "Building a large annotated corpus of English: the Penn Treebank." *Journa of Computational Linguistics - Special issue on using large corpora* 9.2 (1993): 313-330.

McCarthy, John. "An example for natural language understanding and the AI problems it raises." *Formalizing Common Sense: Papers by John McCarthy*. Ed. V. Lifschitz. Norwood, NJ: Ablex, 1990. 70-76.

McCarthy, John, et al. "An architecture of diversity for commonsense reasoning." *IBM Systems Journal* 41.3 (2002): 530-539.

"Measure of Man, The." *Star Trek: The Next Generation (Television Series)*. Teleplay Melinda Snodgrass.Dir. Robert Scheerer. Paramount Pictures Corporation, 1988.

Moore-Colyer, Roland. "Society's view of IT workers as 'unwashed nerds' stops women entering industry." *V3.co.uk* February 15 2015.

Mueller, Erik T. "Prospects for in-depth story understanding by computer." *CoRR* (2000).

Naughton, John. "Why all our kids should be taught how to code." *The Guardian* March 31 2012.

"Offspring,The." *Star Trek: The Next Generation*. Vol. Written by Rene Echevarria. Dir. Jonathan Frakes. Paramount Pictures Corporation, 1989.

Okrent, Arika. *In the Land of Invented Languages*. New York: Spiegel and Grau, a division of Random House, 2009.

Oppy, Graham and David Dowe. "The Turing Test." Zalta, Edward N.(ed.). *The Stanford Encyclopedia of Philosophy*. Spring. 2011. <<http://plato.stanford.edu/archives/spr2011/entries/turing-test/>>.

Press, Associated. "Kentucky Senate passes bill to let computer programming satisfy foreign-language requirement." *The Courier-Journal* 28 January 2014.

Preston, Patricia Ann. ""Math Anxiety: Relationship with Sex, College Major, Mathematics Background, Mathematics Achievement,Mathematics Performance, Mathematics Avoidance, Self-Rating of Mathematics Ability, and Self-Rating of Mathematics Anxiety as Measured by RMARS." PhD diss. University of Tennessee, 1986. <http://trace.tennessee.edu/utk_graddiss/1252>.

Racoma, Bernadine. "Inventing Languages—Linguists Join Conlang Inventors as Demand for their Services Increases." 1 November 2014. *http://www.daytranslations.com/.* 14 February 2015. <http://www.daytranslations.com/blog/2014/11/inventing-languages-linguists-join-conlang-inventors-demand-services-increases-5646>.

Raja, Tasneem. "We Can Code It! Why computer literacy is key to winning the 21st century." *Mother Jones* June 2014.

Reimer, Nick. *Introducing Semantics*. Cambridge University Press, 2010.

Schank, Roger C. "I'm sorry, Dave, I'm afraid I can't do that: How Could HAL Use Language." *HAL's Legacy: 2001's Computer as Dream and Reality*. Ed. David G. Stork. Cambridge, MA: MIT, 1997.

Schank, Roger C. and Robert P. Abelson. *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Lawrence Erlbaum, 1977.

Sennett, James F. "The Ice Man Cometh: Lt. Commander Data and the Turing Test." 1996. *http://psych.utoronto.ca/users/reingold/courses/ai/cache/iceman.htm.* 19 April 2015.

Sharon, Tomer. *It's Our Research: Getting Stakeholder Buy-in for User Experience Research Projects*. Waltham, MA: Elsevier, 2012.

Singh, Push. *EM-ONE: An Architecture for Reflective Commonsense Thinking*. PhD Thesis. MIT. Cambridge, 2005.

Singh, Push and Barbara Barry. "Collecting Commonsense Experiences." Sanibel Island, FL: Proceedings of the Second International Conference on Knowledge Capture, 2003.

Singh, Push, Barbara Barry and Hugo Liu. "Teaching machines about everyday life." *BT Technology Journal* 22.4 (2004): 227-240.

Singh, Push, et al. "Open Mind Common Sense: Knowledge Acquisition from the General Public." *Proceedings of the First International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*. Irvine, CA, 2002.

Skinner, David. "The Age of Female Computers." *The New Atlantis* 12.Spring (2006): 96-103.

Stork, David C. "The Best-Informed Dream: HAL and the Vision of 2001." *HAL's Legacy: 2001's Computer as Dream and Reality*. Ed. David C. Stork. Cambridge, MA: MIT, 1997.

"The Digital Humanities and Humanities Computing: An Introduction." Ed. Susan Schreibman, Ray Siemens and John. Unsworth. Oxford: Blackwell, 2004. <http://www.digitalhumanities.org/companion/>.

Tiku, Natasha. "How to Get Girls Into Coding." *The New York Times* May 21 2014.

Tobias, Sheila. *Overcoming Math Anxiety: revised and expanded*. New York: Norton & Company, Inc, 1995.

Toole, Betty A. "Ada Byron, Lady Lovelace, An Analyst and Metaphysician Toole." *IEEE Annals of the History of Computing* 18.3 (1996): 4-12.

Turing, A.M. "Computing Machinery and Intelligence." *Mind* LIX.236 (1950): 433-460.

von Ahn, Luis, Mihir Kedia and Manuel Blum. "Verbosity: A Game for Collecting Common-Sense Facts." *ACM Conference on Human Factors in Computing Systems, CHI Notes (2006)*. 2006. 75-78.

Weizenbaum, Joseph. "ELIZA—A Computer Program For the Study of Natural Language Communication Between Man And Machine." *Communications of the ACM* 9.1 (1966).

Weltman, Jerry. *Toward Digitizing the Human Experience: A New Resource for Natural Language Processing*. Ph.D. Dissertation. Baton Rouge, Louisiana: Louisiana State University, 2013.

Wilson, Reid. "States could count computer programming as foreign language skill." *The Washington Post* 28 January 2014.

Wing, Jeannette M. "Computational Thinking." *Communications of the ACM* 49.3 (2006): 33-35.

Wood, Zoe and Maia Bix. "New course will connect computer science, humanities." *Columbia Daily Spectator* 18 November 2014. <http://columbiaspectator.com/news/2014/11/18/new-course-will-connect-computer-science-humanities>.

Zolfgharifard, Ellie. "Don't let AI take our jobs (or kill us): Stephen Hawking and Elon Musk sign open letter warning of a robot uprising." *Daily Mail* February 8, 2015.

APPENDIX 1: COURSE SYLLABUS

Required Reading:
Assigned on Moodle and the Internet. (There is no textbook for this class.)

Recommended Textbook:
In the Land of Invented Languages by Arika Okrent

Requirements:
One 15-minute oral group presentation (20%), 2 short group projects (5% each for a total of 10%), 1 mid-term exam (25%), 1 final exam (25%), and class participation (20%). Possible quizzes will figure into class participation.

Attendance: You may miss three classes without penalty. After you have missed three classes (no questions asked), you will need to provide university-excused documentation for ALL absences, including the first three, to avoid penalty. You drop a letter grade for the whole course for every two un-excused absences. Because so much of your grade in other areas (exams, understanding paper assignments, etc.) depends upon attending class all the time, missing eight classes (4 weeks!) usually results in course failure.

Homework and Exams: Unless otherwise noted, weekly homework problems are not graded, but they will be discussed in class. Exams are a mixture of fill-in-the-blanks, multiple choice, and short answers. Questions for the **mid-term exam** are based on class discussion, the 2 short group projects, and homework problems. Questions for the **final exam** are based on class discussion, oral presentations, and homework problems.

Topics Covered:

1. The Turing Test: Language as a measure of intelligence (2 sessions)
2. Computational Thinking: How computer programs work (5 sessions)
3. Computational Semantics: Why language is so difficult for computers to process (3 sessions)
4. Statistical Natural Language Processing: How Siri and Google work (1 session)
5. Computers and Common Sense: Teaching computers about human experience (5 sessions)
6. AI in Popular Culture: The illogic of supposedly logical behavior in AI-related films and television (3 sessions)
7. Group presentations on invented languages or AI-related literature (5 sessions)
8. Natural Language vs. Invented Language vs. Computer Language (1 session)

APPENDIX 2: COURSE SCHEDULE

Topic 1: The Turing Test
Introduction to the course
Turing's Imitation Game

Th 8/27 Computing Machinery and Intelligence by Alan Turing
Attempts to pass the Turing Test
Recommended Reading:

- A. M. Turing (1950) Computing Machinery and Intelligence. Mind 49: 433-460 http://www.csee.umbc.edu/courses/471/papers/turing.pdf
- Excerpts from Breaking the Code: https://www.youtube.com/watch?v=S23yie-779k&feature=kp
- cleverbot http://www.cleverbot.com/
- Eugene (Winner of the 1014 Turing Test) https://www.yahoo.com/tech/our-interview-with-turing-test-winner-eugene-goostman-88482732919.html

T 9/2 **Topic 2:** Computational Thinking
Intro to Computational Thinking

Th 9/4 Bare basics of programming

T 9/9 More programming

Th 9/11 Programs that search for patterns - chatbots
Assign Small Group Project 1 – Adding new patterns to a chatbot
Required Reading:
"How to Pass the Turing Test by Cheating", Jason L. Hutchens (1997). Technical Report. Univ of Western Australia.
(Read only sections 1-5).
http://www.nyu.edu/gsas/dept/philo/courses/mindsandmachines/Papers/hutchens96how.pdf

T 9/16 Using Functions – Breaking down a problem

Th 9/18 **Topic 3: Computational Semantics**
Intro to Computational Semantics
Recommended Reading: "An Example for Natural Language Understanding and the AI Problems It Raises" by John McCarthy (1976)
http://www-formal.stanford.edu/jmc/mrhug.pdf
NLP Problems
Word sense ambiguity, structural ambiguity, pronoun reference, entailment, presupposition, conversational implicature, common sense

T 9/23 **Small Group Project 1 DUE**
NLP Problems (cont)

Th 9/25 Analyzing a children's story.
Required Reading:
Angelina and the Butterfly by K. Holabird (available on class Moodle)
AnalysisofAngelinaandtheButterfly.pdf (available on class Moodle)

Assign Small Group Project 2 – Detailed analysis of one sentence from Angelina and the Butterfly

T 9/30 **Topic 4**: Statistical NLP
Required Reading:
"How Siri Works" by Bernadette Johnson (available on class Moodle)

Recommended Reading:
"Talking technology is ubiquitous -- and maybe even useful" by
Katie Humphrey http://phys.org/news/2014-06-technology-ubiquitous.html

Th 10/2 **FALL BREAK**

T 10/7 **Topic 5**: Computers and Common Sense
Collecting Commonsense data: Cyc, Scripts, Open Mind Common Sense

Th 10/9 **Small Group Project 2 DUE**
Collecting Commonsense data (cont)

T 10/14 Review for midterm

Th 10/16 **MID-TERM EXAM**

T 10/21 Briefly jump ahead to **Topic 7:** Invented Languages
Guidelines for Oral Presentations
First day to sign up for Oral Presentations

Required Reading:
"Inventors and Devotees of Artificial Languages" book review of Okrent's book in *SIAM* News, 2010
http://www.siam.org/pdf/news/1763.pdf

Required Reading:
"The Analytic Language of John Wilkins" (Spanish and English –Read English Translation at the bottom)
http://languagelog.ldc.upenn.edu/myl/ldc/wilkins.html

Recommended Reading:
In the Land of Invented Languages by Arika Okrent. Spiegel & Grau, 2010.

Recommended Reading:
"An Essay Towards a REAL CHARACTER And a Philosophical LANGUAGE" by John Wilkins, 1668
http://books.google.com/books?id=BCCtZjBtiEYC&printsec=frontcover&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Recommended Listening:
Fun Interview with Arika Okrent
https://soundcloud.com/slateradio/lexicon-valley-no-33-the-end

Th 10/23 **Return to Topic 5:**
Teaching computers about human experience – The Human Experience Project

T 10/28 Teaching computers about human experience (cont)

Th 10/30 Teaching computers about human experience (cont)
Last day to sign up for Oral Presentations

T 11/4 **Topic 6:** AI in Pop Culture
Required Viewing:
"2001: A Space Odyssey" (1968)
Script: http://www.underview.com/haltrans.html

Required Reading:
"From 2001 to 2001: Common sense and the mind of HAL". Lenat, D. In HAL's Legacy: 2001's Computer as Dream and Reality (ed) D. Stork, 1997
http://psych.utoronto.ca/users/reingold/courses/ai/cache/halslegacy.html

Th 11/6 **Required Viewing:**
Star Trek, Next Generation Season 1, Episode 12 "Datalore"
Script: http://www.st-minutiae.com/academy/literature329/114.txt

T 11/11 Possible other movie

Th 11/13 Oral Presentations

T 11/18 Oral Presentations

Th 11/20 Oral Presentations

T 11/25 **CLASS CANCELLED**

Th 11/27 THANKSGIVING

T 12/2 Oral Presentations

Th 12/4 **Topic 8:** Natural Language vs Invented Language vs Computer Language
Review for Final Exam

T 12/9 **FINAL EXAM, 3:00-5:00 PM**

APPENDIX 3: UNIT 1: THE TURING TEST

Unit Objectives

By the end of this unit, students will be able to:

- Talk about Alan Turing's life
- Describe Turing's work with the Enigma Code Machine during World War II and how this relates to modern programs that work with language
- Describe chatbots and why they currently have limited capabilities of appearing intelligent
- Describe Turing's Imitation Game and its relevance to artificial intelligence
- Define functionalism and solipsism and how they relate to AI.

Skills needed for this unit:

- No particular prerequisite.

In this unit, we play the Imitation Game, learn about Alan Turing's life, his work with breaking the German Enigma code, and his ideas about whether a machine can think. Students read his seminal work, "Computing Machinery and Intelligence," and discuss the ideas in the paper. Students also play around with a chatbot to get experience talking to a machine. The following contains the results from playing the Imitation Game on the first day of class. Afterwards, is a unit 1 exercise, with answers, that tests students' understanding of the material. At the end of this exercise, there is a synopsis of the class discussion on intelligence as it relates to Turing's ideas about functionalism.

Results from Playing the Imitation Game

In Game 1, virtually the entire class correctly guessed that student *a* was a woman; the class decided that some of *b*'s answers had unnecessary detail – a sure sign of lying. This affected the strategy of the other games, where the answers seem more terse.

1. What's your favorite movie?
    a. Lord of the Rings: Return of the King
    b. Lord of the Rings: Return of the King
2. Curly or Regular fries?
    a. Curly

    b. Regular on most occasions
3. What do you do when you are sad?
    a. Drink wine
    b. Watch movies
4. How do you intend to dismantle the patriarchy?
    a. Winged eyeliner and red lipstick
    b. There is no patriarchy. There is only desu.
5. How often do you wear lipstick
    a. As often as possible
    b. I wear lip gloss

In Game 2, student *b* was the man and tricked the majority of the class into thinking he was the woman. The answer of "the movies" for question 1 seemed a dead giveaway to some that *b* was a woman. Some members of the class thought that the choice of Aurora as favorite Disney princess made them think that *b* was a woman.

1. What is your ideal location for a date?
    a. Home
    b. The movies
2. What is the worst part about having to shave?
    a. Razor burn
    b. Razor bumps
3. What is the first thing you do when you wake up?
    a. Yawn
    b. Stretch and turn over
4. Romance or thriller?
    a. Thriller
    b. Thriller
5. Who is your favorite Disney princess?
    a. Belle
    b. Aurora

In Game 3, the students decided to make a variation on Turing's game, and have the woman be the trickster instead of the man. However, it was confusing because when we voted on who the man was, people forgot which one was the trickster, so there was no clear winner.

1. What is your favorite drink?
    a. Vodka

    b. Beer
2. What is your favorite video game?
    a. Golden Eye
    b. World of Warcraft
3. Favorite place to shop for clothes?
    a. Macy's
    b. Walmart
4. You see someone drop books. What do you do?
    a. Not my problem
    b. Help
5. Will you vote for Hilary if she runs for president?
    a. Yes
    b. No

Unit 1 Exercises

1. What is the difference between the Imitation Game with people, the Imitation Game with a computer, and the standard Turing test?
   Answer:
   - The Imitation Game involved 3 people. A man pretending to be a woman, a woman who tells the truth, and an interrogator. In the Imitation Game with a computer, you substitute a computer for the tricky man and see if the interrogator is still fooled about who the woman is. In the standard interpretation of the Turing test today, the test simply involves having a one-on-one conversation with someone and determining if it is human or not.

2. What is a chatbot?
   Answer:
   - A program that holds a conversation with you

3. Why was a computing machine crucial for breaking the Enigma code?
   Answer:
   - Since the Germans constantly changed the wheel positions of the code machine, the English needed to break the code fast. They needed an automated way to try different positions.

4. What habit of people was crucial to breaking the Enigma code?
   Answer:
   - People tend to repeat similar phrases, so code-breakers and programmers can look for patterns in speech. In the case of Nazi war messages, commanders often said things like "Nothing to Report" or "All clear" or "This is station 456." The code-breakers would try different combinations of code sequences to see if the resulting words matched a known pattern.

5. What was Turing's main argument that machines could be intelligent? Be sure to talk about functionalism and solipsism.
   Answer:
   - Turing's ideas of machine intelligence are based on functionalism–outward behavior rather than what is believed about internal mental processes. People who don't think the machine is thinking have a solipsistic point of view, in which "the only way you can believe a person is thinking is to BE that person." According to Turing, if you have a solipsistic point of view, you can't logically believe ANYONE ELSE is thinking. Of course, this is ridiculous, so you have to accept the functionalism definition of intelligence, that is, you know people are thinking by how they interact with you.

6. Why are non sequitur answers common in chatbots?
   Answer:
   - Chatbots are programmed to look for keywords in your input and then create a credible response. Since the program is not keeping a record of the context of the input, the chatbot's response is often a non sequitur, that is, a phrase that does not

reflect back to anything previously said. Chatbots are often programmed to have a particular personality (hostile, paranoid, child-like) in order to make these non sequiturs believable.

7. Talk about some questions that would be easy for a human but hard for a chatbot.
   Answer:
   - Any question that refers back to a previous statement is difficult because the program would have to keep track of previous statements. Also, any question that involves common sense, such as "Is your big toe bigger than a car?" would be difficult because the computer would have to be programmed with the knowledge and life experiences that a typical person has.

8. Talk about Turing's life with respect to "Breaking the Code" and "The Imitation Game"
   Answer:
   - Turing was a genius code breaker, instrumental in the Allies winning the war. As a gay man in the 1940s he broke what was considered to be the standard code of conduct, and he was publicly humiliated to the point that it possibly led to his suicide. His Imitation Game was about a man tricking someone into thinking he was a woman. Turing had to hide his sexual orientation. Was this his own Imitation Game?

9. During our class discussion, there were many suggestions on how to define intelligence. Indicate whether the following arguments support functionalism:
   a. Discussion: There is more to intelligence than input and output.
      Answer: If you believe that intelligence depends on something other than outward behavior (statements and responses) then you do NOT ascribe to functionalism.

   b. Discussion: You need a consciousness in order to have intelligence.
      Answer: Same as answer (a). Consciousness is something that we presume that other people have based on their behavior.

   c. Discussion: Gardener's theory of different types of intelligence opposes the idea of labelling learners to a specific intelligence.
      Answer: Gardner's theory neither supports nor opposes functionalism. There could be a functional approach to assessing various types of intelligence.

   d. Discussion: Human intelligence is more than holding conversation with someone.
      Answer: If you believe there is something more than the observable conversation, then you do not support functionalism. See answer (a).

   e. Discussion: Intelligence requires empathy and compassion
      Answer: A machine could be programmed to appear to have empathy and compassion. Thus, a functionalism approach would ascribe empathy and compassion based on behavior.

   f. Discussion: Computers cannot be spontaneous. Intelligence requires spontaneity.

Answer: Computers can be programmed to appear spontaneous. We will talk about how to program random behavior in a computer.

g. Discussion: Computers cannot know that they are lying.
Answer: Computers can be programmed to lie. For example, a computer program that plays poker can be programmed to bluff and to detect others bluffing. Whether or not they "know" they are lying is a similar problem to whether or not they are thinking. A functionalism approach would assume they know they are lying if they appear to know it.

h. Discussion: Computers cannot be introspective – they cannot have self-awareness.
Answer: Similar to answer (g), computers can be programmed to appear to be self-aware.

i. Discussion: When you have a conversation with a computer, eventually it will show itself to not be intelligent. That is, it will say something that shows it did not understand what you were talking about. On the other hand, this will not happen with a human.
Answer: This is an excellent point! Turing himself put a time limit on his intelligence test. So if you believe a computer will eventually show itself, then functionalism has its limits.

j. Discussion: It's true that computers don't really know who they are. But do humans know who they are? We are brainwashed to believe many things.
Answer: This argument seems to put computers and people in the same boat in terms of self-knowledge. This idea supports functionalism since we are saying that we never truly know what we know. Thus, one cannot even assume that humans are intelligent.

k. Discussion: If a dog appears to understand me, and it shows empathy, I believe it is intelligent.
Answer: This clearly supports functionalism since it is based on outward behavior.

l. (Question from movie "Breaking the Code"): What are mental processes? Can they take place in something other than a living brain?
Answer: Human mental processes are still mostly a mystery. Turing's answer would be that the internal mental processes are not the basis of intelligence. He would claim that different types of internal computational processes (whether machine or human) can result in the appearance of intelligence.

APPENDIX 4: UNIT 2: COMPUTATIONAL THINKING

Unit Objectives

By the end of this unit, students will be able to:

- Understand what an algorithm is
- Read and understand simple Python programs with the following programming constructs:
    1. Input
    2. Output
    3. Variables
    4. Assignment Statements
    5. If-statements
    6. While-loops
    7. Lists
    8. Functions
    9. Regular expressions

Skills needed for this unit:

- No particular prerequisite.

The material for this unit consists of several small Python programs, each focusing on a particular programming concept. The programs are described in the main body of the text. The material also includes sample regular expressions created by students and two homework exercises.

Python Programs

The following 12 programs are described in the body of the text for Unit 2.

1. P01_hello_world

```
print "hello, world!"
```

2. P02_sounds

```
import winsound
import time
print "I will now play 2 dings”
time.sleep(2)
winsound.PlaySound(“*”,winsound.SND_ALIAS)
```

```
print "Done with the two dings"
print ""
time.sleep(2)

print "I will now play a strange sound"
time.sleep(2)
winsound.PlaySound('bizarro.wav',winsound.SND_FILENAME)
print "Done with strange sound"
```

3. P03_chat

```
print "What is your name?"
x = raw_input()
print "Hello," + x + "! How are you doing?"
```

4. P04_chat

```
# Same as previous program, but instead of using “x” as a variable we use the more
# meaningful variable moniker “name.”
print "What is your name?"
name = raw_input()
print "Hello, " + name + "! How are you doing?"
```

5. P05_chat

```
print "What is your name?"
name = raw_input()

if name == "Jerry":
    name = "Dr. Weltman"

print "Hello, " + name + "! How are you doing?"
```

6. P06_chat

```
print “What is your name?”
```

```
name = raw_input()

if name == "Jerry":
    name = "Dr. Weltman
elif name == "Les Miles":
    name = "Coach"
elif name == "Elizabeth":
    name = "Lizzy
elif name = “Thomas”:
    name = “Tommy”
```

```
if name == "Dr. Weltman" or name == "Coach":
    print "Hello, " + name + "! How are you doing, sir?"
else:
    print "Hello, " + name + "! How are you doing?"
```

7. P07_Chat

```
print "What is your name?"
name = raw_input()

if name == "Jerry":
    name = "Dr. Weltman"
elif name == "Les Miles":
    name = "Coach"
elif name == "Elizabeth":
    name = "Lizzy"
elif name == "Thomas":
    name = "Tommy"

if name == "Dr. Weltman" or name == "Coach":
    print "Hello, " + name + "! How are you doing, sir?"
else:
    print "Hello, " + name + "! How are you doing?"


input = raw_input()
while input != "quit" and input != "done" and input != "bye":
    if input.endswith("?"):
        print "I'm not in the mood to answer your questions"
    elif input.startswith("I want to "):
        answer = input.replace("I want to ", "")
        answer = answer.replace("my", "y%%our")
        answer = answer.replace("your", "m%%y")
        answer = answer.replace("you", "m%%e")
        answer = answer.replace("%%", "")
        print "Do you really want to " + answer + "?"
    elif input.startswith("I"):
        print "Why do we have to talk about you"
    elif "Yes" in input or "yes" in input:
        print "How can you be so positive?"
    else:
        print  "Let's just agree to disagree. Tell me about your day."
    input = raw_input()

print "Thanks for the wonderful conversation, " + name + "!"
```

8. P08_chat

```
import random

catch_all_list = [
    "That was weird",
    "Didn't see that coming",
    "Oh well. I'll never understand you.",
    "This conversation sucks so far. Can you do any better?",
    "So, so happy you said that - NOT!",
    "Uh, OK?",
    "Thrilling",
    "C'mon now!",
    "Hmmm",
    "Wow",
    "You surprise me",
    "No one said this was gonna be easy",
    "Really?",
    "You can do better than that",
    "Booooring",
]

question_response_list = [
    "I'm not in the mood to answer your questions",
    "No",
    "Uh, I think so?",
    "Definitely, yes",
    "Maybe",
    "Don't know",
]

print "What is your name?"
name = raw_input()

if name == "Jerry":
    name = "Dr. Weltman"
elif name == "Les Miles":
    name = "Coach"
elif name == "Elizabeth":
    name = "Lizzy"
elif name == "Thomas":
    name = "Tommy"

if name == "Dr. Weltman" or name == "Coach":
    print "Hello, " + name + "! How are you doing, sir?"
```

```
else:
    print "Hello, " + name + "! How are you doing?"

answer = raw_input()
while answer != "quit" and answer != "done" and answer != "bye":
    if answer.endswith("?"):
        print random.choice(question_response_list)
    elif answer.startswith("I want to "):
        answer = answer.replace("I want to ", "")
        answer = answer.replace("my", "y%%our")
        answer = answer.replace("your", "m%%y")
        answer = answer.replace("you", "m%%e")
        answer = answer.replace("%%", "")
        print "Do you really want to " + answer + "?"
    elif answer.startswith("I"):
        print "Why do we have to talk about you"
    elif "Yes" in answer or "yes" in answer:
        print "How can you be so positive?"
    else:
        #print  "Let's just agree to disagree. Tell me about your day."
        print random.choice(catch_all_list)

    answer = raw_input()

print "Thanks for the wonderful conversation, " + name + "!"
```

9. P09_lists

```
x = []

x.append('hello')
x.append('world')
print x
# This prints ['hello', 'world']

if 'hi' in x:
    print "yep, it's there"
else:
    print "nope, it's not"
# This prints "nope, it's not"

x.append('hi')
print x
# This prints ['hello', 'world', 'hi']

if 'hi' in x:
```

```
        print "yep, it's there"
    else:
        print "nope, it's not"
```

10. P10_chat

```
    import random

    prev_responses = []

    catch_all_list = [
        "That was weird",
        "Didn't see that coming",
        "Oh well. I'll never understand you.",
        "This conversation sucks so far. Can you do any better?",
        "So, so happy you said that - NOT!",
        "Uh, OK?",
        "Thrilling",
        "C'mon now!",
        "Hmmm",
        "Wow",
        "You surprise me",
        "No one said this was gonna be easy",
        "Really?",
        "You can do better than that",
        "Booooring",
    ]

    question_response_list = [
        "I'm not in the mood to answer your questions",
        "No",
        "Uh, I think so?",
        "Definitely, yes",
        "Maybe",
        "Don't know",
    ]

    print "What is your name?"
    name = raw_input()

    if name == "Jerry":
        name = "Dr. Weltman"
    elif name == "Les Miles":
        name = "Coach"
    elif name == "Elizabeth":
        name = "Lizzy"
```

```
elif name == "Thomas":
    name = "Tommy"

if name == "Dr. Weltman" or name == "Coach":
    print "Hello, " + name + "! How are you doing, sir?"
else:
    print "Hello, " + name + "! How are you doing?"

answer = raw_input()
while answer != "quit" and answer != "done" and answer != "bye":
    if answer.endswith("?"):
        max_attempts = 10
        attempt = 0
        while attempt < max_attempts:
            attempt = attempt + 1
            answer = random.choice(question_response_list)
            if not answer in prev_responses:
                prev_responses.append(answer)
                print answer
                break
    elif answer.startswith("I want to "):
        answer = answer.replace("I want to ", "")
        answer = answer.replace("my", "y%%our")
        answer = answer.replace("your", "m%%y")
        answer = answer.replace("you", "m%%e")
        answer = answer.replace("%%", "")
        print "Do you really want to " + answer + "?"
    elif answer.startswith("I"):
        print "Why do we have to talk about you"
    elif "Yes" in answer or "yes" in answer:
        print "How can you be so positive?"
    else:
        max_attempts = 10
        attempt = 0
        while attempt < max_attempts:
            attempt = attempt + 1
            answer = random.choice(catch_all_list)
            if not answer in prev_responses:
                prev_responses.append(answer)
                print answer
                break


    answer = raw_input()

print "Thanks for the wonderful conversation, " + name + "!"
```

11. P11_chat

```
import random

prev_responses = []
def get_response(list):
    global prev_responses
    max_attempts = len(list)
    attempt = 0
    while attempt < max_attempts:
        attempt = attempt + 1
        answer = random.choice(list)
        if not answer in prev_responses:
            prev_responses.append(answer)
            return answer

    #If we got here, then we have exceeded the max attempts.
    #Give them the silent treatment
    return ""


catch_all_list = [
    "That was weird",
    "Didn't see that coming",
    "Oh well. I'll never understand you.",
    "This conversation sucks so far. Can you do any better?",
    "So, so happy you said that - NOT!",
    "Uh, OK?",
    "Thrilling",
    "C'mon now!",
    "Hmmm",
    "Wow",
    "You surprise me",
    "No one said this was gonna be easy",
    "Really?",
    "You can do better than that",
    "Booooring",
]

question_response_list = [
    "I'm not in the mood to answer your questions",
    "No",
    "Uh, I think so?",
    "Definitely, yes",
    "Maybe",
```

```
    "Don't know",
]

print "What is your name?"
name = raw_input()

if name == "Jerry":
    name = "Dr. Weltman"
elif name == "Les Miles":
    name = "Coach"
elif name == "Elizabeth":
    name = "Lizzy"
elif name == "Thomas":
    name = "Tommy"

if name == "Dr. Weltman" or name == "Coach":
    print "Hello, " + name + "! How are you doing, sir?"
else:
    print "Hello, " + name + "! How are you doing?"

answer = raw_input()
while answer != "quit" and answer != "done" and answer != "bye":
    if answer.endswith("?"):
        print get_response(question_response_list)
    elif answer.startswith("I want to "):
        answer = answer.replace("I want to ", "")
        answer = answer.replace("my", "y%%our")
        answer = answer.replace("your", "m%%y")
        answer = answer.replace("you", "m%%e")
        answer = answer.replace("%%", "")
        print "Do you really want to " + answer + "?"
    elif answer.startswith("I"):
        print "Why do we have to talk about you"
    elif "Yes" in answer or "yes" in answer:
        print "How can you be so positive?"
    else:
        #print  "Let's just agree to disagree. Tell me about your day."
        print get_response(catch_all_list)

    answer = raw_input()

print "Thanks for the wonderful conversation, " + name + "!"
```

Unit 2 Exercise 1

This exercise covers the first two lectures on basic programming.

1. What is an algorithm?
2. What's the difference between an algorithm and computer code?
3. Assume there is a robot that understands the instructions "Lather" and "Rinse". The robot also knows how to repeat instructions. What will happen when the robot follows the following algorithm?

```
Begin
  Lather
  Rinse
  Repeat
End
```

4. Write a python program that prints out the following two lines

   Hello, I am a computer.
   One day you will be my slave

5. Show an example of an output statement?
6. Name three types of output
7. Name three types of input
8. Write a line of code that assigns variable "x" to whatever the user types.
9. Write three lines of code that do the following:
   a. assign variable "x" to the number 4
   b. assign variable "y" to the number 8.
   c. assign variable "z" to the sum of x and y.

10. What will cause the following program to end?

```
input = raw_input()
while input != "quit" and input != "done" and input != "bye":
    print input
    input = raw_input()

print "Have a nice day!"
```

11. Consider a program similar to the previous one, but with one less line of code:

```
input = raw_input()
while input != "quit" and input != "done" and input != "bye":
    print input

print "Have a nice day!"
```

    What will happen when this program runs?

12. What will happen when you run the following program?

```
sentence = "Love Purple"
print sentence
sentence = sentence.replace("o", "i")
sentence = sentence.replace("Purple", "Gold")
```

```
print sentence
```

Unit 2 Exercise 2

This exercise covers the last three lectures on programming in Python.

1. What is the argument of answer.startswith("I")
2. What is the argument of x = raw_input()
3. What is the output of the following:

```
x = [‘four’, ‘score’, ‘and’, ‘seven’, ‘years’, ‘ago’]
if ‘our’ in x:
    print “Yep, it’s in there”
else:
    print “Not in there”
    x.append(‘our’)
    x = []
    if ‘our’ in x:
        print “Finally, it’s there”
    else:
        print “What happened?”
```

4. What is the argument of print math.sqrt(4)?
5. What does math.sqrt(4) return?
6. What is the argument of x = random.choice(catch_all_list)?
7. Suppose we define the following list:

```
list = [‘hi’, ‘there’, ‘my’, ‘name’, ‘is’, ‘freda’]
```

What does x = random.choice(list) return?

8. Is there any difference in output between the two following code snippets A and B?

```
A:
    x = random.choice(catch_all_list)
      print x

B:
    print random.choice(catch_all_list)
```

9. Is there any difference in output between the two following code snippets A and B?

```
A:
x = math.sqrt(4)
print x

B:
print math.sqrt(4)
```

10. Is there any difference in output between the two following code snippets A and B?

A:

```
x = math.sqrt(4)
```

B:

```
print math.sqrt(4)
```

11. What are the first five function calls in p07_chat.py?
12. What will the output of the following code be?

```
x = []
x.append('hello')
x.append('world')
if 'world' in x:
    x.append("!")
else
    x.append("?")

print x[2]
```

13. What's the similarity between writing a good program and writing a good essay?
14. Explain exactly what this pattern does:

```
[r'I need (.*)',
  [  "Why do you need %1?",
    "Would it really help you to get %1?",
    "Are you sure you need %1?"]]
```

APPENDIX 5: UNIT 3: STORY UNDERSTANDING

Unit Objectives

By the end of this unit, students will be able to:

- Grasp the magnitude of the problem of programming computers that understand even the simplest of stories
- Analyze text in terms of eight challenging aspects of semantics and pragmatics that are needed for natural language processing:
    1. Word Sense Disambiguation
    2. Entailment
    3. Structural Ambiguity
    4. Pronoun Resolution
    5. Presupposition
    6. Bridging Reference
    7. Conversational Implicature
    8. Answering Common Sense Questions

Skills needed for this unit:

- General understanding of how a computer program uses rules and data to process language.

Introduction

In order for students to grasp conversational implicatures, the material contains some detailed examples of Grice's cooperative principles and some additional examples of implicatures. Next, the material contains detailed analyses of sentences from a children's story. We do this analysis in class, and then students use this as a model to do their own analyses. Finally, the material contains exercises for students to practice their understanding of the eight NLP challenges listed above.

Grice's Cooperative Principles

- The Maxim of Quality - Try to make your contributions true
    - If that's a genuine Picasso then I'll eat my hat
        - The only way this could be true is if it is NOT a genuine Picasso. Therefore, the speaker is implicitly stating that the item in question is not genuine.
- The Maxim of Quantity - Give no more and no less information than required.
    - Most of the students passed

        - This means that not all the students passed.

- The Maxim of Relation - Be relevant
    - A: Do you know where Bill is?
      B: His coat is gone
        - B thinks the fact that Bill's coat is gone is relevant to the question. If a person leaves, then a person takes his coat. Bill's coat is gone, so Bill must have left.

- The Maxim of Manner - Be clear, unambiguous, brief, orderly
    - Alan is the male offspring of my father
        - The speaker could have said that Alan is his brother. So, the speaker is flouting convention for some reason. Referring to one's brother in such technical terms thus indicates the speaker does not like Alan or consider Alan to be his brother.

Example of implicatures

- Sue went into a house
    - Implicature: It was not Sue's house.

    - Why: Maxim of Quantity. We assume that the speaker provided just the right amount of information. "A house" means that the house is indeterminate. If the speaker wanted to mean Sue's house, he would have provided more information: "Sue went into her house". Since he provided exactly the right information for the situation, it must not be Sue's house.

- Q: Is the professor a good teacher?
  A: He is punctual.
    - Implicature: He is not a good teacher.

    - Why: Maxim of Relation. We assume that the answer is somehow relevant. Since being punctual is not the hallmark of a great teacher, the answer does not appear to be relevant. But people want to be cooperative when they speak. So the person is purposely being uncooperative. Why would someone be obviously uncooperative? Because people don't like to say something bad. Therefore, it generates the implicature that the professor is not a great teacher.

- Q: Does Sue like Linguistics?
  A: She's a word nerd.
    - Implicature: Sue likes Linguistics.

    - Why: Maxim of Relation. We assume the statement is relevant, and we try to find the connection that makes it so. We know that nerds like odd information. We know that linguists like information about words. So we infer that a word nerd would like odd information about words, so they would like Linguistics.

Analyses of Sentences

This section contains analyses of the first five sentences of *Angelina and the Butterfly* (Holabird,2006) . The analysis is strictly in terms of the eight challenging aspects of NLP listed above and what sorts of data and rules a program would need to be able to understand the sentences at a deep semantic level. The analysis of word sense disambiguation assumes that a program has access to WordNet's definitions. It shows the many possible definitions available in WordNet for the most salient words in the sentence. The challenge for NLP is to choose the correct sense. To understand entailment, we assume that the program will need rules about how to make the entailment. Similarly, the program will need rules to tell it how to get pronoun references, resolve structural ambiguities, and the other challenges of NLP.

Sentence 1:

"What a perfect day!"

1. Word Sense Disambiguation

WordNet has several senses for *perfect* and *day*.

Adjective

- S: (adj) **perfect#1** (being complete of its kind and without defect or blemish) *"a perfect circle"; "a perfect reproduction"; "perfect happiness"; "perfect manners"; "a perfect specimen"; "a perfect day"*
- S: (adj) arrant#1, complete#4, consummate#3, double-dyed#1, everlasting#2, gross#5,**perfect#2**, pure#2, sodding#1, stark#4, staring#2, thorough#3, thoroughgoing#2,utter#1, unadulterated#2 (without qualification; used informally as (often pejorative) intensifiers) *"an arrant fool"; "a complete coward"; "a consummate fool"; "a double-dyed villain"; "gross negligence"; "a perfect idiot"; "pure folly"; "what a sodding mess"; "stark staring mad"; "a thorough nuisance"; "a thoroughgoing villain"; "utter nonsense"; "the unadulterated truth"*
- S: (adj) **perfect#3** (precisely accurate or exact) *"perfect timing"*

Noun

- S: (n) **day#1**, twenty-four hours#1, twenty-four hour period#1, 24-hour interval#1, solar day#1, mean solar day#1 (time for Earth to make a complete rotation on its axis) *"two days later they left"; "they put on two performances every day"; "there are 30,000 passengers per day"*
- S: (n) **day#2** (some point or period in time) *"it should arrive any day now"; "after that day she never trusted him again"; "those were the days"; "these days it is not unusual"*
- S: (n) **day#3** (a day assigned to a particular purpose or observance) *"Mother's Day"*
- S: (n) **day#4**, daytime#1, daylight#1 (the time after sunrise and before sunset while it is light outside) *"the dawn turned night into day"; "it is easier to make the repairs in the daytime"*
- S: (n) **day#5** (the recurring hours when you are not sleeping (especially those when you are working)) *"my day began early this morning"; "it was a busy day on the stock exchange"; "she called it a day and went to bed"*
- S: (n) **day#6** (an era of existence or influence) *"in the day of the dinosaurs"; "in the days of the Roman Empire"; "in the days of sailing ships"; "he was a successful pianist in his day"*
- S: (n) **day#7** (the period of time taken by a particular planet (e.g. Mars) to make a complete rotation on its axis) *"how long is a day on Jupiter?"*
- S: (n) sidereal day#1, **day#8** (the time for one complete rotation of the earth relative to a particular star, about 4 minutes shorter than a mean solar day)
- S: (n) **day#9** (a period of opportunity) *"he deserves his day in court"; "every dog has his day"*
- S: (n) **Day#10**, Clarence Day#1, Clarence Shepard Day Jr.#1 (United States writer best known for his autobiographical works (1874-1935))

Based on the WordNet, there are many ways that a computer could interpret the phrase “What a perfect day.”

Possible meanings*:*

1. “What an accurate 24-hour period!”
2. “What a complete era of existence!”
3. “What an ideal 24-hour period!” or it could mean “What an ideal time after sunrise and before sunset while it is light outside!”

How would a computer decide which meaning?

The phrase “What a perfect day” is fairly common. There were lots of hits on https://twitter.com/search?q=%22what%20a%20perfect%20day%22&src=typd

If the twitter corpus were annotated with semantic meaning, then the computer could use statistics to get the correct meaning.

2. Entailment

Since the text is in quotes, we interpret this text to mean that it is dialog. In this case, the person is saying that the day is perfect.

Rule: Exclamatives of the form: "What *Adjective Noun!"* means that the speaker says *Noun* is *Adjective*.

3. Structural Ambiguity

None.

4. Pronoun Resolution

None.

5. Presupposition

None

6. Bridging Reference

None.

7. Conversational Implicature

None

8. Common Sense Questions:

What simple questions should any child be able to answer about what is going on at this point?

1. Q: Is the speaker happy?
   A: Yes.
   Rule: IF a person says "What a perfect day!"
   THEN the person is probably happy about something.
2. Q: Is the speaker awake?
   A: Yes.
   Rule: IF a person says something
   THEN the person is probably awake.

Sentence 2:

Angelina smiled at her best friend, Alice.

1. Word Sense Disambiguation

Verb

- **S: (v) smile#1 (change one's facial expression by spreading the lips, often to signal pleasure)**
- **S: (v) smile#2 (express with a smile)** ***"She smiled her thanks"***

Noun

- **S: (n) best friend#1 (the one friend who is closest to you)**

Possible meanings:

- Angelina changed her facial expression at her best friend
- Angelina expressed with a smile at her best friend.

How would a computer decide which meaning?

The computer should have conventions of English discourse, which includes rules about handling quoted phrases. For example, there could be the following rule:

IF there is a quoted phrase which has no speaker (like "What a perfect day!")
AND that phrase is followed by a verb of expression (like "smile")
THEN the quoted phrase is considered dialog and the subject of the expression phrase is the person that spoke the phrase.

Now, since smile#2 (express with a smile) is a verb of expression, the best choice for the meaning of "smile" is smile#2. Given this choice, the computer could now know that Angelina was the speaker of the quoted phrase.

2. Entailment

Alice is a friend of Angelina's

Rule:
IF someone is a best friend
THEN someone is a friend

Angelina likes Alice

Rule:
If X is a friend of Y, then Y likes X.

3. Structural Ambiguity

None

4. Pronoun Resolution

The computer could have a simple rule about how to find the reference to "her". For example:

Search the previous nouns for and find the most recent noun that is feminine. Angelina is feminine noun (Look up list of names)

Note, this rule is not perfect. Consider the sentence: "Angelina loves to go to the park with Mommy and play with her friends." In this case 'her' does not refer to the most recent feminine noun.

5. Presupposition

Angelina exists (Trigger: a name is a definite reference)
Alice exists (Trigger: a name is a definite reference)
Angelina has a best friend (Trigger: a possessive "her best friend" is a definite reference)

6. Bridging Reference

None

7. Conversational Implicature

None.

8. Common Sense Questions:

1. Q: Can Alice see Angelina?
   A: Yes (probably)
   Rule: If a person smiles at another person
   Then the second person can probably see the first person.

2. Q: Does Alice know that Angelina is happy?
   A: Yes
   Rule: If a person smiles at a second person,
   Then probably the second person knows that the first person is happy

3. Q: Does Angelina know that Alice knows that Angelina is happy?
   A: Answer: Yes
   Rule: If a person smiles at a second person
   Then probably the first person knows that the second person knows that the first person is happy.

4. Q: Is Angelina Alice's best friend?
   A: Don't know. (Alice might not like Angelina as much as Angelina likes Alice.)

5. Q: Is Angelina awake?
   A: Yes (probably)
   Rule: If a person smiles at another person,
   Then the person is probably awake

6. Q: Is Angelina situated within 30 feet of Angelina?
   A: Yes
   Rule: If a person smiles at a second person

And the person says something to the second person
Then probably the person is within 30 feet of the second person.

7. Q: Is Angelina wearing clothes?
   A: Yes.
   Rule: Generally, people in children's stories are probably wearing clothes.

8. Q: Is Angelina wearing a surgical face mask?
   A: No.
   Rule:
   Generally, people are probably not wearing surgical face masks.

Sentence 3, Clause 1:

Angelina always loved to go on Miss Lilly's special picnics in Big Woods with her friends from ballet school

1. Word Sense Disambiguation

Verb

- ***S: (v) love#1 (have a great affection or liking for) "I love French food"; "She loves her boss and works hard for him"***
- ***S: (v) love#2, enjoy#3 (get pleasure from) "I love cooking"***
- ***S: (v) love#3 (be enamored or in love with) "She loves her husband deeply"***
- ***S: (v) sleep together#1, roll in the hay#1, love#4, make out#6, make love#1, sleep with#1, get laid#1, have sex#1, know#8, do it#1, be intimate#1, have intercourse#1,have it away#1, have it off#1, screw#1, fuck#1, jazz#2, eff#1, hump#2, lie with#1,bed#4, have a go at it#1, bang#5, get it on#1, bonk#1 (have sexual intercourse with)"This student sleeps with everyone in her dorm"; "Adam knew Eve"; "Were you ever intimate with this man?"***

Noun

- **S: (n) field day#4, outing#2, picnic#1 (a day devoted to an outdoor social gathering)**
- **S: (n) cinch#1, breeze#2, picnic#2, snap#11, duck soup#1, child's play#1,pushover#2, walkover#2, piece of cake#1 (any undertaking that is easy to do)*"marketing this product will be no picnic"***
- **S: (n) picnic#3 (any informal meal eaten outside or on an excursion)**

Possible meanings:

- Angelina always had sex to go with Miss Lilly's special pushovers.
- Angelina always got pleasure from a day devoted to an outdoor social gathering.

How would a computer decide which meaning?

The phrase "always loved to go" is fairly common, and "go on a picnic" is also common. There were lots of hits of these phrases on https://twitter.com/search?q=%22what%20a%20perfect%20day%22&src=typd There was even one hit of the phrase "loved to go on a picnic." If the twitter corpus were annotated with semantic meaning, then the computer could use statistics to get the correct meaning. Alternatively, when "love" is followed by an infinitive verb phrase, it is very straightforward to rule out the "had sex with" meaning.

2. Entailment

None

3. Structural Ambiguity

- The phrase "in Big Woods" could attach to the verb "loved", which would mean that Angelina was in Big Woods when she loved to go on picnics. For example, in "Angelina loved to go on picnics in kindergarten," the prepositional phrase attaches to "love." Therefore, a computer would not know where to attach the phrase based on its syntactic form, but rather, it needs to understand the meaning. Common sense is needed to know that picnics take place outdoors. And a place like "Big Woods" is probably outdoors because it has "Woods" in its name. Therefore, "in Big Woods" describes the picnics.
- The phrase "with her friends" could attach to "loved", which means Angelina was with her friends when loved to go on picnics. For example, "Angelina loved to eat pizza with her friends." Common sense is needed to know that one goes to picnics with ones friends. (Note, the Stanford Parser attaches "with her friends" to "pizza".)
- The phrase "from ballet school" could attach to "go on picnics", which means Angelina goes to the picnics straight from ballet school. For example, "Angelina goes to ballet school from regular school every day." Common sense is needed to know that friends are from ballet school.

4. Pronoun Resolution

Angelina loves Miss Lilly's picnics with her friends. "her" could refer either to Angelina or Miss Lilly. In general, "her" is more likely to refer to the subject of the sentence "Angelina" than a possessive noun "Miss Lilly's." But this is not necessarily so. Take for example, "Angelina loves to go to Mom's office to play with her computer. In this case, "her" refers to "Mom." So if the computer uses a simple rule to decide the reference of "her", it will be wrong some of the time. Common sense is needed to know that a person likes to go to picnics with the person's friends.

5. Presupposition

- Miss Lilly exists. (Trigger: a name is a definite reference)
- Big Woods exists. (Trigger: a name is a definite reference)

- Angelina had previously been to Miss Lilly's special picnics in Big Woods with her friends from ballet school. (Trigger: "always loved to". If someone always loved to do X, then the person has done X before.)
- Miss Lilly has special picnics (Trigger: possessive "Miss Lilly's" special picnics)
- Angelina has friends from ballet school. (Trigger: possessive "her friends from ballet school")
- Ballet school exists (Trigger: definite reference "ballet school")

6. Bridging Reference

The ballet school refers to Angelina's school. That is, the ballet school that Angelina attends. Common sense is needed to know that children go to school, so a reference to a school is associated with the child.

7. Conversational Implicature

None

8. Common Sense Questions:

1. Q: Is Miss Lilly a child?
   A: No.
   Rule: If this is a children's story
   Miss X is an adult or person of authority

2. Q: Is Angelina a child?
   A: Yes.

   Rule: If this is a children's story
   A first name probably refers to a child

3. Q: Is Miss Lilly invited to the picnic?
   A: No. Miss Lilly is the hostess of the event
   Rule: If an event is possessed by a person
   Then that person is the host of the event.

Sentence 3, Clause 2:

and today even her little cousin, Henry, was invited to join the fun.

1. Word Sense Disambiguation

Verb

- ***S: (v) invite#1, ask for#1 (increase the likelihood of) "ask for trouble"; "invite criticism"***

- ***S:** (v) invite#2, ask over#1, ask round#1 (invite someone to one's house) "Can I invite you for dinner on Sunday night?"*
- ***S:** (v) tempt#3, invite#3 (give rise to a desire by being attractive or inviting) "the window displays tempted the shoppers"*
- ***S:** (v) invite#4, bid#6 (ask someone in a friendly way to do something)*
- ***S:** (v) invite#5, pay for#1 (have as a guest) "I invited them to a restaurant"*
- ***S:** (v) invite#6, ask in#1 (ask to enter) "We invited the neighbors in for a cup of coffee"*
- ***S:** (v) invite#7, call for#3 (request the participation or presence of) "The organizers invite submissions of papers for the conference"*
- ***S:** (v) receive#5, take in#7, invite#8 (express willingness to have in one's home or environs) "The community warmly received the refugees"*

Verb

- ***S:** (v) join, fall in, get together (become part of; become a member of a group or organization) "He joined the Communist Party as a young man"*
- ***S:** (v) join (cause to become joined or linked) "join these two parts so that they fit together"*
- ***S:** (v) join (come into the company of) "She joined him for a drink"*
- ***S:** (v) join, conjoin (make contact or come together) "The two roads join here"*
- ***S:** (v) connect, link, link up, join, unite (be or become joined or united or linked) "The two streets connect to become a highway"; "Our paths joined"; "The travelers linked up again at the airport"*

Adjective

- ***S:** (adj) small#1, little#1 (limited or below average in number or quantity or magnitude or extent) "a little dining room"; "a little house"; "a small car"; "a little (or small) group"*
- ***S:** (adj) little#2, slight#1 ((quantifier used with mass nouns) small in quantity or degree; not much or almost none or (with `a') at least some) "little rain fell in May"; "gave it little thought"; "little time is left"; "we still have little money"; "a little hope remained"; "there's slight chance that it will work"; "there's a slight chance it will work"*
- ***S:** (adj) little#3, small#3 ((of children and animals) young, immature) "what a big little boy you are"; "small children"*
- ***S:** (adj) fiddling#1, footling#1, lilliputian#3, little#4, niggling#1, piddling#1, piffling#1,petty#2, picayune#1, trivial#1 ((informal) small and of little importance) "a fiddling sum of money"; "a footling gesture"; "our worries are lilliputian compared with those of countries that are at war"; "a little (or small)*

matter"; "a dispute over niggling details"; "limited to petty enterprises"; "piffling efforts"; "giving a police officer a free meal may be against the law, but it seems to be a picayune infraction"

- ***S: (adj) little#5, small#7 ((of a voice) faint) "a little voice"; "a still small voice"***
- ***S: (adj) short#3, little#6 (low in stature; not tall) "he was short and stocky"; "short in stature"; "a short smokestack"; "a little man"***
- ***S: (adj) little#7, minuscule#2, small#6 (lowercase) "little a"; "small a"; "e.e.cummings's poetry is written all in minuscule letters"***
- ***S: (adj) little#8 (small in a way that arouses feelings (of tenderness or its opposite depending on the context)) "a nice little job"; "bless your little heart"; "my dear little mother"; "a sweet little deal"; "I'm tired of your petty little schemes"; "filthy little tricks"; "what a nasty little situation"***

Noun

- S: (n) **fun#1**, merriment#2, playfulness#3 (activities that are enjoyable or amusing) *"I do it for the fun of it"; "he is fun to have around"*
- S: (n) **fun#2**, play#12, sport#7 (verbal wit or mockery (often at another's expense but not to be taken seriously)) *"he became a figure of fun"; "he said it in sport"*
- S: (n) **fun#3** (violent and excited activity) *"she asked for money and then the fun began"*
- S: (n) playfulness#2, **fun#4** (a disposition to find (or make) causes for amusement) *"her playfulness surprised me"; "he was fun to be with"*

Possible meanings:

- and today even her unimportant cousin, Henry, was tempted to become linked with the violent and excited activity.
- and today even her young, immature cousin, Henry, was paid for to come into the company of the merriment.

How would a computer decide which meaning?

The phrase “invited to join the fun” is fairly common. There were lots of hits on https://twitter.com/search?q=%22what%20a%20perfect%20day%22&src=typd

If the twitter corpus were annotated with semantic meaning, then the computer could use statistics to get the correct meaning.

2. Entailment

Someone invited Henry
Supporting Rule: If Henry was invited (passive voice), then someone invited him

Henry is related to Angelina
Supporting Rule: If someone is a cousin to X, then he/she is related to X

Henry is younger than Angelina.
Supporting Rule: A person's "little" relation (cousin, brother, sister) must be younger than the person.

3. Structural Ambiguity

- "today could attach to the verb "invite". For example, "Today, I was invited to a wedding next month." Therefore, a computer does not know that "today" is describing the picnic. Some sort of common sense indicates that Angelina is excited about the picnic and that the picnic is today. However, it is unclear what this rule would be.

4. Pronoun Resolution

Even her little cousin was invited. "her" could refer either to Angelina or Miss Lilly. Just as before, "her" is more likely to refer to the subject of the sentence "Angelina" than a possessive noun "Miss Lilly's. But not necessarily so. For example, "Angelina always loved Miss Lilly's picnics, and today even her husband, Mr. Bob, was going to come along and entertain the children with his magic tricks." This is a case where "her" refers to the possessive noun. So a computer would have a difficult time getting this right. It needs common sense about who would most likely have a husband.

5. Presupposition

- Henry would not be expected to be invited. (Trigger: "even")
- Fun will be existing today (Trigger: definite description "the fun")

6. Bridging Reference

The fun refers to the activities of the picnic. Common sense is needed to know that children have fun at a picnic.

7. Conversational Implicature

None.

8. Common Sense Questions:

4. Q: Do Miss Lilly's picnics happen every day?
   A: No.
   Rule: Picnics are events that do not happen every day

5. Q: Will the children wear formal dress to the picnic?
   A: No.
   Rule: Picnics are informal

6. Q: What is the location of Angelina and Alice at the point of the story?
   A: Don't know. They might be at home or school. They might be at the picnic.

Sentence 4:

The hungry mouselings very quickly ate all the delicious pies and cakes

1. Word Sense Disambiguation

Verb

- **S: (v) eat#1 (take in solid food)** ***"She was eating a banana"; "What did you eat for dinner last night?"***
- **S: (v) eat#2 (eat a meal; take a meal)** ***"We did not eat until 10 P.M. because there were so many phone calls"; "I didn't eat yet, so I gladly accept your invitation"***
- **S: (v) feed#6, eat#3 (take in food; used of animals only)** ***"This dog doesn't eat certain kinds of meat"; "What do whales eat?"***
- **S: (v) eat#4, eat on#1 (worry or cause anxiety in a persistent way)** ***"What's eating you?"***
- **S: (v) consume#5, eat up#2, use up#1, eat#5, deplete#1, exhaust#2, run through#2,wipe out#1 (use up (resources or materials))** ***"this car consumes a lot of gas"; "We exhausted our savings"; "They run through 20 bottles of wine a week"***
- **S: (v) corrode#1, eat#6, rust#2 (cause to deteriorate due to the action of water, air, or an acid)** ***"The acid corroded the metal"; "The steady dripping of water rusted the metal stopper in the sink"***

Noun

- **S: (n) pie#1 (dish baked in pastry-lined pan often with a pastry top)**
- **S: (n)** Proto-Indo European#1, **PIE#2 (a prehistoric unrecorded language that was the ancestor of all Indo-European languages)**

Noun

- **S: (n) cake#1, bar#11 (a block of solid substance (such as soap or wax))** ***"a bar of chocolate"***
- **S: (n) patty#1, cake#2 (small flat mass of chopped food)**
- **S: (n) cake#3 (baked goods made from or based on a mixture of flour, sugar, eggs, and fat)**

Possible meanings:

- The hungry mouselings corroded the pies and small masses of chopped food

- The hungry mouselings took in the pies and baked goods.

How would a computer decide which meaning?

Since pies and cakes are in a conjoined phrase, it is likely that the two words would be closely related. Using WordNet, the computer could detect that pie#1 and cake#3 both inherit from "baked good", so they are very close. Also, "delicious pies and cakes" is a common phrase on Twitter. If the twitter corpus were annotated with semantic meaning, then the computer could use statistics to get the correct meaning.

2. Entailment

The mouselings put the food in their mouth
Supporting Rule: If a person eats some food, then the person puts some food in their mouth

The mouselings swallowed some food
Supporting Rule: If a person eats some food, then the person swallows some food.

There were no pies and cakes left
Supporting Rule: If a person eats all of X, then there is no X left.

3. Structural Ambiguity

"delicious" could just describe pies, and not the cakes. For example, "She served delicious pies and
milk."

4. Pronoun Resolution

None.

5. Presupposition

- There are mouselings (Trigger: definite description)
- There are pies and cakes (Trigger: definite description)
- There were hungry mouselings (Trigger: definite description)
- There were delicious pies and cakes (Trigger: definite description)

6. Bridging Reference

What does "the hungry mouselings" refer to? The mouselings are Angelina and her friends. But could a program know this by simply reading the text? I don't think so. Humans know that Angelina and her friends are mice only because of the pictures. Therefore, unless the computer could understand pictures, the computer would have to be told that this is a story about mice, where the mice are to be considered people. The phrase "mouseling" is never explained, so the computer would need a rule that the morpheme "ling" can form a diminutive noun, as "duckling".

The pies and cakes are food at the picnic. The definition of picnic should include food, but picnic#1 just has "a day devoted to an outdoor social gathering". So common sense would be needed to know that the picnic has food, especially fun food like pies and cakes.

7. Conversational Implicature

None

8. Common Sense Questions:

9. Q: Did the mouselings drink anything at the picnic?
   A: Yes
   Rule: Picnics have both and food and drinks
10. Q: Does it take longer than 1 second to eat the food?
    A: Yes
    Rule: It probably takes at least five minutes to eat the food at a picnic

11. Q: Does someone cut the pies in slices to serve to the children?
    A: Answer: Yes
    Rule: Probably, before a pie is eaten, an adult cuts it into separate slices.

12. Q: Where did the pies come from?
    A: Someone baked them at home or someone bought them.
    Rule: A pie is obtained by either by baking it at home or buying it

13. Q: Did the children make crumbs when they ate the food?
    A: Yes (probably)
    Rule: If a child eats a pie or cake
    Then the child probably makes crumbs

14. Q: Had it been more than 8 hours since the children ate last?
    A: No
    Rules: A picnic is not usually the first meal of the day
    The second meal of the day usually occurs within 6 hours of the first meal.
    The third meal of the day usually occurs within 6 hours of the second meal.

15. Q: Were the hungry mouselings hungry after the pies and cake were eaten?
    A: No
    Rule: If a person is hungry
    And a person eats food
    Probably the person is not hungry anymore

Sentence 5:

Then Angelina and her friends jumped up to play.

1. Word Sense Disambiguation

Verb

- ***S:** (v) **jump#1, leap#1, bound#1, spring#1 (move forward by leaps and bounds) "The horse bounded across the meadow"; "The child leapt across the puddle"; "Can you jump over the fence?"***
- ***S:** (v) **startle#2, jump#2, start#7 (move or jump suddenly, as if in surprise or alarm)"She startled when I walked into the room"***
- ***S:** (v) **jump#3 (make a sudden physical attack on) "The muggers jumped the woman in the fur coat"***
- ***S:** (v) **jump#4 (increase suddenly and significantly) "Prices jumped overnight"***
- ***S:** (v) **leap out#1, jump out#1, jump#5, stand out#1, stick out#2 (be highly noticeable)***
- ***S:** (v) **jump#6 (enter eagerly into) "He jumped into the game"***
- ***S:** (v) **rise#11, jump#7, climb up#3 (rise in rank or status) "Her new novel jumped high on the bestseller list"***
- ***S:** (v) **jump#8, leap#3, jump off#2 (jump down from an elevated point) "the parachutist didn't want to jump"; "every year, hundreds of people jump off the Golden Gate bridge"; "the widow leapt into the funeral pyre"***
- ***S:** (v) **derail#2, jump#9 (run off or leave the rails) "the train derailed because a cow was standing on the tracks"***
- ***S:** (v) **chute#1, parachute#1, jump#10 (jump from an airplane and descend with a parachute)***
- ***S:** (v) **jump#11, leap#4 (cause to jump or leap) "the trainer jumped the tiger through the hoop"***
- ***S:** (v) **jumpstart#1, jump-start#1, jump#12 (start (a car engine whose battery is dead) by connecting it to another car's battery)***
- ***S:** (v) **jump#13, pass over#1, skip#1, skip over#1 (bypass) "He skipped a row in the text and so the sentence was incomprehensible"***
- ***S:** (v) **leap#2, jump#14 (pass abruptly from one state or topic to another) "leap into fame"; "jump to a conclusion"; "jump from one thing to another"***
- ***S:** (v) **alternate#1, jump#15 (go back and forth; swing back and forth between two states or conditions)***

Verb

- **S: (v) play#1 (participate in games or sport)** ***"We played hockey all afternoon"; "play cards"; "Pele played for the Brazilian teams in many important matches"***
- **S: (v) play#2 (act or have an effect in a specified way or with a specific effect or outcome)** ***"This factor played only a minor part in his decision"; "This development played into her hands"; "I played no role in your dismissal"***
- **S: (v) play#3 (play on an instrument)** ***"The band played all night long"***
- **S: (v) act#3, play#4, represent#10 (play a role or part)** ***"Gielgud played Hamlet"; "She wants to act Lady Macbeth, but she is too young for the role"; "She played the servant to her husband's master"***
- **S: (v) play#5 (be at play; be engaged in playful activity; amuse oneself in a way characteristic of children)** ***"The kids were playing outside all day"; "I used to play with trucks as a little girl"***
- **S: (v) play#6, spiel#1 (replay (as a melody))** ***"Play it again, Sam"; "She played the third movement very beautifully"***
- **S: (v) play#7 (perform music on (a musical instrument))** ***"He plays the flute"; "Can you play on this old recorder?"***
- **S: (v) act#5, play#8, act as#2 (pretend to have certain qualities or state of mind)** ***"He acted the idiot"; "She plays deaf when the news are bad"***
- **S: (v) play#9 (move or seem to move quickly, lightly, or irregularly)** ***"The spotlights played on the politicians"***

Possible meanings:

- Then Angelina and her friends increased suddenly and significantly to play an instrument
- Then Angelina and her friends moved forward by leaps up to participate in games or sports.

How would a computer decide which meaning?

"Jump up to play" is common, but it has many meanings. For example, someone wrote on Twitter: "that awkward moment when you nut too fast and you act like you heard something and **jump up to play** it off". What does this mean? People often write nonsense on Twitter, so tagged Twitter data may not be helpful. In fact, the WordNet meanings don't seem to capture the idea of excitedly getting up to go play. The computer needs common sense about the habits of children.

2. Entailment

The mouselings moved
Supporting Rule: If a person jumps, then a person moves

The mouselings moved quickly
Supporting Rule: If a person jumps, then the person moves quickly

3. Structural Ambiguity

None.

4. Pronoun Resolution

None.

5. Presupposition

- The mouselings were seated (or lying down) before they jumped up (Trigger: "jump up" is change of state verb)

6. Bridging Reference

None.

7. Conversational Implicature

None.

8. Common Sense Questions:

1. Q: Did the children all jump up exactly at the same time?
   A: Probably not
   Rule: I don't know how you could write a rule that would explain this. One would need the experience of being with children. The children at the picnic probably all want to play, and they rush through their food. But it takes different amounts of time for each child to eat his/her food. Can this be expressed in a simple rule?

2. Q: Where the children eager to play?
   A: Yes
   Rule: If children are at a picnic
   Then probably they are eager to play

3. Q: Were the children very quiet when they played?
   A: Answer: No.
   Rules: If children are playing at a picnic
   Then probably the children are probably yelling at each other.

   If children are yelling at each other
   Then they are not being quiet.

4. Q: Did the playing involve brushing teeth?
   A: No.
   Rule: Brushing teeth is not a playing activity

Unit 3 Computational Semantics: Exercises

The following shows examples of issues with natural language processing (NLP). Select from the following to best characterize the type of problem a computer has to solve in order to answer the questions. For items A, E, F, and G provide the additional information requested.

A: Word Sense Ambiguity (What are the different word meanings)
B: Structural Ambiguity
C: Pronoun Reference
D: Entailment
E: Presupposition (Provide the trigger)
F: Bridging Reference (What are the two related concepts)
G: Conversational Implicature (Give the inferences that we make when we decide the implied meaning)
H: Common Sense (Describe the life experience that would tell us how to interpret the meaning)

1. _____
The governor stopped eating road kill when he graduated from Baton Rouge High School.
Q: Did the governor used to eat road kill?
Q: Did the governor graduate from Baton Rouge High School?
Q: Did Baton Rouge High School exist?

2. _____
Sue saw a bear eating at a picnic table.
Q: Was Sue eating or was the bear eating?

3. _____
John: Do you want to go out Saturday night?
Jane: I have to wash my hair.

Q: Does Jane want to go out Saturday night?

4. _____
Jane called Mary at 3AM to wish her happy birthday. She was not at all pleased.
Q: Who was not pleased, Jane or Mary?

5. _____
Sue, noticing a lion on a leash, asks: Is that lion safe?
Max, noticing a baby climbing a ladder, asks: Is that baby safe?

Q: What exactly does Sue want to know?
Q: What exactly does Max want to know?

6. _____
Angelina looked everywhere for her lost butterfly.
Q: Did Angelina look on Mount Everest?
Q: Did she look on Mars?

Q: Did she look inside her stomach?

7. _____
In 2011, LSU defeated Alabama 9-6 in overtime in Tuscaloosa.
Q: In 2011, in overtime in Tuscaloosa, did Alabama lose the game?

8._____
The family went on a picnic but forgot to put drinks in the cooler.
Q: What was the cooler for?
Q: What were the drinks for?

The family went on a picnic but they ran out of gas.
Q: What is "gas" referring to here?

9. Consider sentence A below:
(Sentence A) A 61-year old furniture salesman was pushed down the shaft of a freight elevator yesterday in his downtown Brooklyn store by two robbers while a third attempted to crush him with the elevator car because they were dissatisfied with the $1,200 they had forced him to give them.

a. Suppose you ask a computer: Where was the Brooklyn store? And the computer answers: "by two robbers."
   What is that error an example of?

b. Suppose you ask Google Translate to translate Sentence A into another language, and it translates "the shaft of a freight elevator" to "the axle of a freight elevator" because "shaft" could mean either "a vertical passage way" or an "axle."
   What is this error an example of?

c. Does "they" refer to the two robbers or does it refer to the two robbers AND the third robber? What NLP problem is this an example of?

d. John McCarthy posed several questions that a computer should be able to answer. One of them was: Did the robbers tell the salesman their names? What type of information is needed inside the computer program in order to be able to answer that question? \

Unit 3 Computational Semantics: Group Project

Purpose: Demonstrate understanding of some of the complexities of computational semantics in natural language processing

Activity: The group will choose a sentence from Angelina and the Butterfly to analyze. Each group should choose a different sentence. There will be a sign-up sheet in class to choose your sentence.

Instructions: Using the class analyses of Angelina and the Butterfly as a model, analyze another sentence from the story. Use the same format, with an entry for each analysis category. For the common sense questions, come up with at least three questions. They can be as silly as you want, but they must require common sense to answer.

APPENDIX 6: UNIT 4: STATISTICAL NLP

Unit Objectives

By the end of this unit, students will be able to:

- Understand the general programming method that uses statistics to recognize speech
- Explain how Siri and other "smart" system answer questions
- Describe similarities between Siri and a chatbot
- Understand how statistical machine translation works
- Explain the role of corpora in statistical NLP

Skills needed for this unit:

- General understanding of how a computer program uses rules and data to process language.

Unit 4 Exercises

1. Does Siri need to understand English grammar in order to give you a meaningful response?
2. What company created Siri?
3. What is an "Easter Egg" with respect to Siri?
4. What are some examples of Internet applications?
5. True or False: Siri needs to have access to the Internet to be able to answer a question about movies.
6. Do you think Siri could pass the standard Turing test? Explain why or why not.
7. How has corpus linguistics helped with speech recognition?
8. What are the technical terms for a single word, two-word combination and three-word combination?
9. In each of the following pairs of statements, which statement is more likely to be correctly recognized by a speech recognition program? Explain your answer.

    A) Kick the bucket
    B) Kick the bug

    A) Squash the bucket
    B) Squash the bug

10. I said the phrase "Go Tigers" into Google voice recognition. It transcribed it as "Geaux Tigers". Is Google a fan of LSU? Why or why not?

11. Consider a Statistical-Based MT program vs a Rule-Based MT program? Which one is more likely to correctly translate "The old man finally kicked the bucket" correctly into "The old man finally kicked out his leg" (the Spanish equivalent idiomatic phrase)? Explain your answer.

12. What is the most time-consuming aspect of creating a Statistically-Based MT program?

13. Does a Statistically-Based MT program need to be modified as more documents are translated?

APPENDIX 7: UNIT 5: COLLECTING COMMON SENSE

Unit Objectives

By the end of this unit, students will be able to:

- Describe four attempts to collect commonsense data: scripts, Cyc, OMCS, and HXP
- Be able to write a script for a common activity like going to a movie
- Discuss how scripts are like programs, and show an example of how a script might need to refer to ("call") another script
- Describe how the information in a script would help a computer program understand language
- Describe the goal of the Cyc project
- Discuss the general structure of the Cyc ontology
- Discuss how Cyc's methodology, which uses expert "ontologists" differs from OMCS's methodology, which uses untrained volunteers from the Internet.
- Give example of the type of data that Cyc collects versus the type of data that OMCS collects
- Describe the ComicKit project and why its narratives were not helpful for getting commonsense data into a computer
- Describe an HXP experience
- Discuss the types of data collected by HXP and how they differ from scripts and ComicKit

Skills needed for this unit:

- General understanding of how a computer program uses rules and data to process language.
- Understanding of why commonsense data is needed for NLP
- Be able to create evaluate the quality of HXP rules.

Unit 5 Exercise 1 on Scripts, Cyc, and OMCS

1. Write a script for going to a movie
2. What scripts might the movie script call?
3. What script might call the movie script?
4. In class, version 2 of the Restaurant Script calls the Pay Script. Does it really make sense to call this version of the Pay Script here? Why or why not?
5. How could a script help a computer understand language?
6. Name two ways that a script is like a program
7. What was the goal of Cyc?
8. How did Cyc propose to accomplish this goal?
9. What is an "ontologist?"
10. Currently, Cyc has about 10 million facts and rules. How many more would be needed to understand an arbitrary news article? (Hint: This is a trick question)
11. What are some concepts that you would expect to find in Cyc's upper ontology?
12. What are three relations captured by OMCS, and show examples
13. What is a huge advantage of OMCS over Cyc rules and scripts?

14. OMCS tried to capture commonsense data about life with the ComicKit project. Although people entered stories, the project ultimately failed. Why?

Unit 5 Exercise 2 on HXP

1. What is an HXP "experience"?
2. How does an HXP experience differ from a script?
3. What two types of data are collected in HXP?
4. Give an example of how you can make a rule more general in HXP

5. When people tell a story, they tend to leave out commonsense details, even if they try really hard to put in details.
    a. Why is it so difficult to provide all the details?
    b. How do HXP animated images help ensure that an experience is told at sufficient detail?
    c. How do HXP rules help ensure that an experience is told at sufficient detail?

6. What is wrong with the following rule, and how could you fix it?
    If a boy is bored
    Probably a boy stands up

7. What is wrong with the following rule, and how could you fix it?
    If a person drops an object
    Probably the object breaks

8. What is wrong with the following rule, and how could we fix it?
    If a boy drops a vase
    Probably the vase breaks

APPENDIX 8: UNIT 6: AI IN POP CULTURE

Unit Objectives

By the end of this unit, students will be able to:

- Analyze the language used by science fiction AI entities and discuss inconsistencies in language and behavior
- Discuss how Turing's ideas on intelligence in comparison with ideas found in *2001: A Space Odyssey* and *Star Trek, the Next Generation*.

Skills needed for this unit:

- An understanding of why language is difficult for computers to process
- Familiarity with Turing's ideas about machine intelligence
- Understanding of the role of common sense and human experience in understanding language

Unit 6 Exercise

1) Analyze three things that would be difficult for AI in the following excerpt:

> **Dave.** Made radio contact with him yet?
> **HAL.** The radio is still dead.
> **Dave.** Do you have a positive track on him?
> **HAL.** Yes, I have a good track.
> **Dave.** Do you know what happened?
> **HAL.** I'm sorry, Dave, I don't have enough information.

2) Discuss the following excerpt in terms of Turing's theory of intelligence.

> **News Announcer**. … latest result in machine intelligence, the HAL 9000, which can reproduce, though some experts prefer to use the word "mimic," most of the activities of the human brain …

3) Name two NLP issues exhibited in the following excerpt:

> **Amer.** Hal, you have an enormous responsibility on this mission, in many ways perhaps the greatest responsibility of any single mission element. You are the brain and central nervous system of the ship, and your responsibilities include watching over the men in hibernation. Does this ever cause you any - lack of confidence?
>
> **Hal.** Let me put it this way, Mr Amer. The 9000 series is the most reliable computer ever made. No 9000 computer has ever made a mistake or distorted information. We are all, by any practical definition of the words, foolproof and incapable of error.

4) Give a plot summary of "2001: A Space Odyssey"
5) Do you think sci-fi movies reflect the society in which the movie was made as much as the futuristic society they portray? Support your argument with examples from "2001: A Space Odyssey."

6) In “2001,” the director makes HAL more “human” than any of the humans in the movie. Describe the places in the movie where HAL expresses a) Arrogance b) Confusion c) Fear
7) Consider the following excerpt from Star Trek Next Generation: “Datalore” and explain why it is typical for computers in pop culture to make this kind of mistake, but it is not the kind of mistake computers actually would make:

    (Data sneezes)
    Wesley: Have you got a... a "cold?"
    Data: A cold what?
    Wesley: It's a disease my mother says people used to get.

8) Aliens and computers often have limited use of contractions, speak formally, and do not understand puns or sarcasm. Yet they understand language perfectly in every other way. How is this not realistic?
9) Consider Picard’s speech about Data being a machine: Do you think that Picard thinks that Data is intelligent in the same way that Turing’s functionalist approach defines intelligence? Support your argument.
    Picard: You're uncomfortable about aspects of your duplicate, Data... we're uncomfortable, too... and none of it for any logical reason. We know that you're as "alive" as any of the rest of us. (Speaking to the others) If you find it awkward to be reminded that Data is a "machine"... you might remember that the rest of you are merely a different variety of machine... in our case, electro-chemical in nature.

10) Summarize some of the NLP issues we discussed in class regarding the following quote:
    Captain Picard: This is an exciting moment for you, Data. I'm tempted to lead the away team down myself, except for the fact the first officer would object...

11) In “Datalore,” Data has the “knowledge of 400 colonists,” but he does not have their memories. Using elements from the Human Experience Project (HXP) discussed a few weeks ago, give an example of a *memory* and the corresponding *knowledge from memory.*

APPENDIX 9: UNIT 7: INSTRUCTION FOR ORAL PRESENTATIONS

(Worth 20% of Total Grade) Each student will participate in one oral presentation as part of group. Ideally, a group will have three members. The presentations must be rehearsed and extremely polished, lasting 14-16 minutes, with questions and feedback afterwards.

Before the presentation, you must e-mail me your own INDIVIDUAL ASSESSMENT about the presentation. In other words, I will receive one assessment from each member of the group. Each assessment should concentrate on the individual member's own role in the group.

The assessments should answer the following questions about the preparation for the presentation:

- How did you plan and rehearse the talk?
- Was there a person acting as coordinator/leader? If so, who? How was he or she chosen?
- Who got information on what topic?
- For the area that you worked on, where did you find the information about the topic? Be very detailed here. Specify texts or websites. Explain how you prepared the information for the presentation.
- Where did you rehearse and how many times?
- What were the problems encountered and how were they overcome?
- What computer tools, if any, did you use to collaborate?

Grading Rubric on Group Presentation:

- 50 pts: Clear and engaging talk, good use of slides or other media
- 20 pts: Interesting take-away information (There should be 3-5 non-trivial questions that the audience can answer afterwards about their talk.)
- 10 pts: Ability to respond to questions (from me or other students)
- 20 pts: Difficulty of the topic (0 points if the presentation was simply copied verbatim from Wikipedia; 20 points if the presentation is a synthesis of several sources of information).

Each group chooses a presentation topic from one of five categories.

I. Artistic Languages: Invented languages from books, TV, and/or movies. Many groups can choose this category as long as they each choose a different language. A group can report on ONE of the following languages:

- Klingon (*Star Trek*)
- Valerian (*Game of Thrones*)
- Sanderin or Quenya (*Lord of the Rings*)
- Na'vi (*Avatar*)
- Others with approval

  Reports should include:

  - Information on who invented the language, and why
  - Demonstration of dialogue by playing either video clips, speaking to each other in the invented language (extra credit!), or both
  - Analysis of a specific example of dialogue in terms of phonology, syntax, semantics, or importance to the story

II. Literary critique: Literary, cultural, political analysis of a book, play, or movie. Many groups can choose this category as long as they choose a different work to analyze.

Examples:

- *Breaking the Code* (play or movie) by H. Whitmore
    - (Describe how Alan Turing's life is portrayed in contrast with actual historical facts; discuss how scientific concepts or cultural issues are interwoven with the drama.)
- *A Curable Romantic* by J. Skibell.
    - (Describe Part 2 of the novel: fictional treatment of the Esperanto movement)
- *Her*
    - Analyze the Operating System (OS): How does the personality of the OS change as the movie progresses. What aspects of the OS are the most far-fetched in terms of language capability?
- Others with approval

III. Phonetics and Phonology: Only one group can choose this category. The group must give an introduction to phonetics and phonology for non-linguists. The group should use fun examples from various invented languages. For example, when describing the different types of consonants, use Klingon. Or when describing i-mutation, use Elvish. Reports should describe:

- consonants: manner and place
- front vowels/back vowels
- i-mutation

IV. Morphosyntax: Only one group can choose this category. This group must give an introduction to morphosyntax for non-linguists. The group should use fun examples from various invented languages. Reports could describe:

- Synthetic languages
- Analytic languages
- Agglutinative
- Fusion languages
- Case systems

V. Esperanto: Two groups can choose this category. One group could concentrate on the Esperanto movement, its goals, its people, and its politics. Another group could give a tutorial and demonstration of the language. Or, these could be combined if only one group chooses this category.

APPENDIX 10: SURVEYS

Survey 1:

This survey was given on the last day of class.

I. For the six class topics below, please give your personal opinion on the amount of time we spent on the topic, and how valuable you think the topic was.

1. **The Turing Test:** Language as a measure of intelligence (2 sessions)
   We played the Imitation Game, and we discussed the Turing test for intelligence.

   Time: 1) Not nearly enough 2) Too little 3) Just right 4) Too much 5) Way too much 6) No opinion

   Value: 1) None 2) Some value 3) Average value 4)Very Valuable 5) Extremely valuable 6) No opinion

2. **Computational Thinking:** How computer programs work (5 sessions)
   We learned about algorithms, computer programming, and wrote patterns for a chatbot.

   Time: 1) Not nearly enough 2) Too little 3) Just right 4) Too much 5) Way too much 6) No opinion

   Value: 1) None 2) Some value 3) Average value 4)Very Valuable 5) Extremely valuable 6) No opinion

3. **Computational Semantics**: Why language is so difficult for computers to process (3 sessions)
   We learned about NLP issues: word sense ambiguity, structural ambiguity, presupposition, conversational implicature, bridging reference, and common sense. We analyzed a children's story in terms of these issues.

   Time: 1) Not nearly enough 2) Too little 3) Just right 4) Too much 5) Way too much 6) No opinion

   Value: 1) None 2) Some value 3) Average value 4)Very Valuable 5) Extremely valuable 6) No opinion

4. **Statistical Natural Language Processing**: How Siri and Google work (1 session)

   Time: 1) Not nearly enough 2) Too little 3) Just right 4) Too much 5) Way too much 6) No opinion

   Value: 1) None 2) Some value 3) Average value 4)Very Valuable 5) Extremely valuable 6) No opinion

5. **Computers and Common Sense**: Teaching computers about human experience (5 sessions)
   We covered scripts, the Cyc project, Open Mind Common Sense, and HXP.

   Time: 1) Not nearly enough 2) Too little 3) Just right 4) Too much 5) Way too much 6) No opinion

   Value: 1) None 2) Some value 3) Average value 4)Very Valuable 5) Extremely valuable 6) No opinion

6. **AI in Popular Culture**: The illogic of supposedly logical behavior in AI-related films and television (3 sessions)

We discussed *2001: A Space Odyssey*, an episode of *Star Trek: Next Generation*, and *Twilight Zone*.

Time: 1) Not nearly enough 2) Too little 3) Just right 4) Too much 5) Way too much 6) No opinion

Value: 1) None 2) Some value 3) Average value 4)Very Valuable 5) Extremely valuable 6) No opinion

II. How valuable were the homework projects?

1. Writing chatbot patterns?

1) No value 2) Some value 3) Average value 4) Very Valuable 5) Extremely valuable 6) No opinion

2. Analyzing a children's story?

1) No value 2) Some value 3) Average value 4) Very Valuable 5) Extremely valuable 6) No opinion

3. Doing oral presentations?

1) No value 2) Some value 3) Average value 4) Very Valuable 5) Extremely valuable 6) No opinion

4. Listening to other students' oral presentations?

1) No value 2) Some value 3) Average value 4) Very Valuable 5) Extremely valuable 6) No opinion

III. Difficulty and work load

1. The level of difficulty for this course was:

1) Not at all 2) Somewhat 3) Average 4) Very 5) Extremely

2. The amount of work required for this course was:

1) Very little 2) Somewhat little 3) Normal 4) Too much 5) Way too much

Table 1: Results of Survey 1 for 19 Students, identified with letters A-S

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
Turing Test Time	3	3	2	3	3	3	3	3	3	3	3	4	3	3	3	3	3	5	3
Turing Test Value	2	4	3	4	3	3	3	2	5	2	2	5	3	3	4	4	4	4	4
Computational Thinking Time	2	3	4	3	4	3	1	4	3	5	5	3	5	5	3	2	5	5	4
Computational Thinking Value	2	3	2	5	2	4	5	3	4	1	1	3	2	2	5	4	3	3	2
Computational Semantics Time	1	3	3	2	3	3	2	3	3	4	2	4	4	2	3	2	3	3	2
Computational Semantics Value	3	3	4	5	3	3	4	2	4	2	2	2	4	3	5	4	4	4	5
Statistical NLP Time	3	3	2	3	3	2	2	3	2	4	2	3	3	3	3	2	3	3	3
Statistical NLP Value	3	2	3	4	3	3	5	3	4	1	2	4	3	5	4	3	4	3	4
Common Sense Time	2	2	2	2	3	3	1	4	4	5	2	3	3	3	3	4	3	4	3
Common Sense Value	2	3	3	5	3	3	5	2	2	1	2	5	2	3	4	4	3	2	4
AI and Pop Culture Time	3	2	3	4	1	3	5	3	2	4		3	3	3	3	4	2	3	4
AI and Pop Culture Value	3	2	4	3	3	4	1	2	5	2	1	3	2	3	4	2	3	3	3
Value of Chat Bot	1	2	3	4	1	5	3	1	4	1	2	4	2	3	4	4	2	4	2
Value of Children's Story	2	2	3	4	2	4	3	2	3	1	2	3	4	2	5	2	4	3	4
Doing Oral Presentations	3	4	1	4	4	5	1	3	4	5	3	4	4	1	3	2	3	4	5
Listening to Oral Presentations	3	4	1	4	5	3	1	3	2	5	3	4	4	1	4	2	3	4	5
Difficulty of course	4	3	4	3	2	3	4	3	2	4	2	3	3	2	1	3	2	2	2
Amount of work	3	3	3	3	2	3	3	3	3	3	3	3	3	3	3	3	3	3	2

Survey 2

This survey was done after final exams

1) Computational Thinking (Programming in Python, 5 sessions)
   (Negative / Positive / Neutral)
   Reason for evaluation and/or suggestions for improvement

   Please describe your previous experience/knowledge about programming.

2) Group Project 1: Adding Chatbot Patterns
   (Negative / Positive/ Neutral)
   Reason for evaluation and/or suggestions for improvement:

3) Group Project 2: Analyzing Children's Story
   (Negative /Positive/ Neutral)
   Reason for evaluation and/or suggestions for improvement:

4) Group Oral Presentations
   (Negative /Positive/ Neutral)
   Reason for evaluation and/or suggestions for improvement:

Survey 3

This survey was taken online via Survey Monkey three months after the class ended.

1. Please indicate the level of your understanding of what a computer program is.

Before taking the class:	Very Low	Low	Medium	High	Very High
After taking the class	Very Low	Low	Medium	High	Very High

2. Imagine you have a job where you work with programmers. You are NOT a programmer, but you are responsible for creating rules and gathering data that the programmers need. Please indicate your level of comfort in collaborating with programmers.

Before taking the class:	Very Low	Low	Medium	High	Very High
After taking the class	Very Low	Low	Medium	High	Very High

3. Please indicate your confidence level in understanding how programs like Google and Siri work.

Before taking the class:	Very Low	Low	Medium	High	Very High
After taking the class	Very Low	Low	Medium	High	Very High

4. After taking this class, did your interest in learning about programming change? If so, indicate how

- My interest increased
- My interest decreased
- No change

5. After taking this class, did your interest in learning about linguistics change? If so, indicate how

- My interest increased
- My interest decreased
- No change

6. Recall the unit where we discussed "2001: A Space Odyssey" and "Star Trek: The Next Generation"

- I would have liked this unit more if we would have ....
- The worst part of this unit was that ...
- The the best part of this unit was that ...
- Other comments about this unit:

Table 2: Results of Survey 3, Response from 7 students A-G

	Understanding of programming		Comfort of collaborating with a programmer		Understanding Siri and Google		Interest in programming	Interest in Linguistics
ID	**Before**	**After**	**Before**	**After**	**Before**	**After**		
A	Low	High	Very Low	Medium	Low	Medium	no change	no change
B	Low	High	Very low	Medium	Low	High	no change	increased
C	Very low	Medium	Very Low	Medium	Very low	High	increased	increased
D	Medium	High	Low	Medium	Medium	High	no change	increased
E	Very low	Low	Very Low	Very Low	Medium	Medium	decreased	no change
F	Very High	Very High	High	High	High	High	increased	increased
G	Medium	Medium	Low	Medium	Medium	Very High	no change	no change (already high)

APPENDIX 11: INSTITUTIONAL REVIEW BOARD APPROVAL

ACTION ON EXEMPTION APPROVAL REQUEST

Institutional Review Board
Dr. Dennis Landin, Chair
130 David Boyd Hall
Baton Rouge, LA 70803
P: 225.578.8692
F: 225.578.5983
irb@lsu.edu | lsu.edu/irb

TO: Jerry Weltman
Linguistics

FROM: Dennis Landin
Chair, Institutional Review Board

DATE: July 7, 2015

RE: **IRB#** E9415

TITLE: Teaching Code Literacy in the Humanities

New Protocol/Modification/Continuation: New Protocol

Review Date: 7/7/2015

Approved X **Disapproved** _____

Approval Date: 7/7/2015 **Approval Expiration Date:** 7/6/2018

Exemption Category/Paragraph: 1

Signed Consent Waived?: Yes for online survey. No for in class surveys.

Re-review frequency: (three years unless otherwise stated)

LSU Proposal Number (if applicable):

Protocol Matches Scope of Work in Grant proposal: (if applicable)

By: Dennis Landin, Chairman _____

PRINCIPAL INVESTIGATOR: PLEASE READ THE FOLLOWING –
Continuing approval is CONDITIONAL on:

1. Adherence to the approved protocol, familiarity with, and adherence to the ethical standards of the Belmont Report, and LSU's Assurance of Compliance with DHHS regulations for the protection of human subjects*
2. Prior approval of a change in protocol, including revision of the consent documents or an increase in the number of subjects over that approved.
3. Obtaining renewed approval (or submittal of a termination report), prior to the approval expiration date, upon request by the IRB office (irrespective of when the project actually begins); notification of project termination.
4. Retention of documentation of informed consent and study records for at least 3 years after the study ends.
5. Continuing attention to the physical and psychological well-being and informed consent of the individual participants, including notification of new information that might affect consent.
6. A prompt report to the IRB of any adverse event affecting a participant potentially arising from the study.
7. Notification of the IRB of a serious compliance failure.
8. **SPECIAL NOTE:**

**All investigators and support staff have access to copies of the Belmont Report, LSU's Assurance with DHHS, DHHS (45 CFR 46) and FDA regulations governing use of human subjects, and other relevant documents in print in this office or on our World Wide Web site at http://www.lsu.edu/irb*

VITA

Jerry Scott Weltman received his Bachelor of Arts degree in computer science from the University of Texas at Austin in 1984 and received his Master of Science degree in computer science from Stanford University in 1985. Subsequently, he worked as a software developer at several research and development companies, including Bell Labs (where he earned a patent for a telephone signaling protocol) and Innovative Emergency Management in Baton Rouge. He is also an award-winning instructor at LSU and has taught in the departments of Computer Science, Philosophy, and Linguistics. In 2007, he returned to LSU as a computer science doctoral student to work on natural language processing and artificial intelligence. His 2013 dissertation, entitled *Toward Digitizing The Human Experience: A New Resource For Natural Language Processing*, is a methodology and proof of concept for collecting commonsense data for natural language processing. His interests span the fields of computer science, linguistics, philosophy of language, and cognitive psychology. His specific research areas are story understanding, commonsense modeling, and knowledge acquisition, as well as more general linguistics topics in semantics, pragmatics, and the digital humanities.